



INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

(Approved by AICTE | NAAC Accreditation with 'A' Grade | Accredited by NBA | Affiliated to JNTUH)

Dundigal, Hyderabad - 500 043, Telangana

**OUTCOME BASED EDUCATION
WITH
CHOICE BASED CREDIT SYSTEM**

**BACHELOR OF TECHNOLOGY
ELECTRICAL AND ELECTRONICS ENGINEERING**

**ACADEMIC REGULATIONS, COURSE CATALOGUE and SYLLABUS –
BT23**

**B.Tech Regular Four Year Degree Program
(for the batches admitted from the academic year 2023 - 2024)**

&

**B.Tech (Lateral Entry Scheme)
(for the batches admitted from the academic year 2024 - 2025)**

**These rules and regulations may be altered / changed from time to time by the academic council
FAILURE TO READ AND UNDERSTAND THE RULES IS NOT AN EXCUSE**

VISION

To bring forth professionally competent and socially sensible engineers, capable of working across cultures meeting the global standards ethically.

MISSION

To provide students with an extensive and exceptional education that prepares them to excel in their profession, guided by dynamic intellectual community and be able to face the technically complex world with creative leadership qualities.

Further, be instrumental in emanating new knowledge through innovative research that emboldens entrepreneurship and economic development for the benefit of wide spread community.

QUALITY POLICY

Our policy is to nurture and build diligent and dedicated community of engineers providing a professional and unprejudiced environment, thus justifying the purpose of teaching and satisfying the stake holders.

A team of well qualified and experienced professionals ensure quality education with its practical application in all areas of the Institute.

PROGRAM OUTCOMES (PO's)

Engineering Graduates will be able to:

- PO1: Engineering Knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
- PO2: Problem Analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
- PO3: Design/Development of Solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
- PO4: Conduct Investigations of Complex Problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
- PO5: Modern Tool Usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
- PO6: The Engineer and Society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
- PO7: Environment and Sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
- PO8: Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
- PO9: Individual and Team Work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
- PO10: Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
- PO11: Project Management and Finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
- PO12: Life-Long Learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

CONTENTS

Section	Particulars	Page
1	Choice Based Credit System	1
2	Medium of Instruction	1
3	Programs Offered	1
4	Semester Structure	2
5	Registration / Dropping / Withdrawal	3
6	Credit System	3
7	Curricular Components	4
8	Evaluation Methodology	5
9	Attendance Requirements and Detention Policy	12
10	Conduct of Semester End Examinations and Evaluation	13
11	Scheme for the Award of Grade	13
12	Letter Grades and Grade Points	13
13	Computation of SGPA and CGPA	14
14	Illustration of Computation of SGPA and CGPA	15
15	Review of SEE Theory Answer Books	16
16	Promotion Policies	16
17	Graduation Requirements	17
18	Award of Degree	17
19	B.Tech with Honours or Minor in Engineering	18
20	Temporary Break of Study from the Program	21
21	Termination from the Program	21
22	With-holding of Results	22
23	Graduation Day	22
24	Discipline	22
25	Grievance Redressal Committee	22
26	Transitory Regulations	22
27	Revision of Regulations and Curriculum	23
28	Frequently asked Questions and Answers about autonomy	24
29	Malpractice Rules	28
30	Course Catalog	31
31	Undertaking by Student / Parent	492

**“Take up one idea.
Make that one idea your life-think of it, dream of it, live on that idea.
Let the brain muscles, nerves, every part of your body be full of that idea and just leave every other idea
alone. This is the way to success”**

Swami Vivekananda

PRELIMINARY DEFINITIONS AND NOMENCLATURES

AICTE: Means All India Council for Technical Education, New Delhi.

Autonomous Institute: Means an institute designated as Autonomous by University Grants Commission (UGC), New Delhi in concurrence with affiliating University (Jawaharlal Nehru Technological University, Hyderabad) and State Government.

Academic Autonomy: Means freedom to an institute in all aspects of conducting its academic programs, granted by UGC for Promoting Excellence.

Academic Council: The Academic Council is the highest academic body of the institute and is responsible for the maintenance of standards of instruction, education and examination within the institute. Academic Council is an authority as per UGC regulations and it has the right to take decisions on all academic matters including academic research.

Academic Year: It is the period necessary to complete an actual course of study within a year. It comprises two main semesters i.e., (one odd + one even) and one supplementary semester.

Branch: Means specialization in a program like B.Tech degree program in Aeronautical Engineering, B.Tech degree program in Computer Science and Engineering etc.

Board of Studies (BOS): BOS is an authority as defined in UGC regulations, constituted by Head of the Organization for each of the departments separately. They are responsible for curriculum design and updation in respect of all the programs offered by a department.

Backlog Course: A course is considered to be a backlog course, if the student has obtained a failure grade (F) in that course.

Basic Sciences: The courses offered in the areas of Mathematics, Physics, Chemistry etc., are considered to be foundational in nature.

Betterment: Betterment is a way that contributes towards improvement of the students' grade in any course(s). It can be done by either (a) re-appearing or (b) re-registering for the course.

Commission: Means University Grants Commission (UGC), New Delhi.

Choice Based Credit System: The credit based semester system is one which provides flexibility in designing curriculum and assigning credits based on the course content and hours of teaching along with provision of choice for the student in the course selection.

Certificate Course: It is a course that makes a student to have hands-on expertise and skills required for holistic development in a specific area/field.

Compulsory course: Course required to be undertaken for the award of the degree as per the program.

Continuous Internal Examination: It is an examination conducted towards sessional assessment.

Core: The courses that are essential constituents of each engineering discipline are categorized as professional core courses for that discipline.

Course: A course is offered by a department for learning in a particular semester.

Course Outcomes: The essential skills that need to be acquired by every student through a course.

Credit: A credit is a unit that gives weight to the value, level or time requirements of an academic course. The number of 'Contact Hours' in a week of a particular course determines its credit value. One credit is equivalent to one lecture/tutorial hour per week.

Credit point: It is the product of grade point and number of credits for a course.

Cumulative Grade Point Average (CGPA): It is a measure of cumulative performance of a student over all the completed semesters. The CGPA is the ratio of total credit points secured by a student in various courses in all semesters and the sum of the total credits of all courses in all the semesters. It is expressed up to two decimal places.

Curriculum: Curriculum incorporates the planned interaction of students with instructional content, materials, resources, and processes for evaluating the attainment of Program Educational Objectives.

Department: An academic entity that conducts relevant curricular and co-curricular activities, involving both teaching and non-teaching staff, and other resources in the process of study for a degree.

Detention in a Course: Student who does not obtain minimum prescribed attendance in a course shall be detained in that particular course.

Dropping from Semester: Student who doesn't want to register for any semester can apply in writing in prescribed format before the commencement of that semester.

Elective Course: A course that can be chosen from a set of courses. An elective can be Professional Elective and / or Open Elective.

Evaluation: Evaluation is the process of judging the academic performance of the student in her/his courses. It is done through a combination of continuous internal assessment and semester end examinations.

Experiential Engineering Education (ExEEd): Engineering entrepreneurship requires strong technical skills in engineering design and computation with key business skills from marketing to business model generation. Our students require sufficient skills to innovate in existing companies or create their own.

Grade: It is an index of the performance of the students in a said course. Grades are indicated by alphabets.

Grade Point: It is a numerical weight allotted to each letter grade on a 10 - point scale.

Honours: An Honours degree typically refers to a higher level of academic achievement at an undergraduate level.

Institute: Means Institute of Aeronautical Engineering, Hyderabad unless indicated otherwise by the context.

Massive Open Online Courses (MOOC): MOOC courses inculcate the habit of self learning. MOOC courses would be additional choices in all the elective group courses.

Minor: Minor are coherent sequences of courses which may be taken in addition to the courses required for the B.Tech degree.

Pre-requisite: A specific course, the knowledge of which is required to complete before student register another course at the next grade level.

Professional Elective: It indicates a course that is discipline centric. An appropriate choice of minimum number of such electives as specified in the program will lead to a degree with specialization.

Program: Means, UG degree program: Bachelor of Technology (B.Tech); PG degree program: Master of Technology (M.Tech) / Master of Business Administration (MBA).

Program Educational Objectives: The broad career, professional and personal goals that every student will achieve through a strategic and sequential action plan.

Project work: It is a design or research-based work to be taken up by a student during his/her final year to achieve a particular aim. It is a credit based course and is to be planned carefully by the student.

Re-Appearing: A student can reappear only in the semester end examination for theory component of a course to the regulations contained herein.

Registration: Process of enrolling into a set of courses in a semester of a program.

Regulations: The regulations, common to all B.Tech programs offered by Institute, are designated as "BT23" and are binding on all the stakeholders.

Semester: It is a period of study consisting of 16 weeks of academic work equivalent to normally minimum of 90 working days. Odd semester commences usually in July and even semester in December of every year.

Semester End Examinations: It is an examination conducted for all courses offered in a semester at the end of the semester.

S/he: Means “she” and “he” both.

Student Outcomes: The essential skill sets that need to be acquired by every student during her/his program of study. These skill sets are in the areas of employability, entrepreneurial, social and behavioral.

University: Means Jawaharlal Nehru Technological University Hyderabad (JNTUH), Hyderabad, is an affiliating University.

Withdraw from a Course: Withdrawing from a course means that a student can drop from a course within the first two weeks of odd or even semester (deadlines are different for summer sessions). However, s/he can choose a substitute course in place of it, by exercising the option within 5 working days from the date of withdrawal.

PREFACE

Dear Students,

The focus at IARE is to deliver value-based education with academically well qualified faculty and infrastructure. It is a matter of pride that IARE continues to be the preferred destination for students to pursue an engineering degree.

In the year 2015, IARE was granted academic autonomy status by University Grants Commission, New Delhi under Jawaharlal Nehru Technology University Hyderabad. From then onwards, our prime focus is on developing and delivering a curriculum which caters to the needs of various stakeholders. The curriculum has unique features enabling students to develop critical thinking, solve problems, analyze socially relevant issues, etc. The academic cycle designed on the basis of Outcome Based Education (OBE) strongly emphasizes continuous improvement and this has made our curriculum responsive to current requirements.

The curriculum at IARE has been developed by experts from academia and industry and it has unique features to enhance problem solving skills apart from academic enrichment. The curriculum of B.Tech program has been thoroughly revised as per AICTE / UGC / JNTUH guidelines and have incorporated unique features such as competency training / coding, industry driven elective, internship and many more. The curriculum is designed in a way so as to impart engineering education in a holistic approach towards Excellence.

I hope you will have a fruitful stay at IARE.

Dr. L V Narasimha Prasad
Principal



INSTITUTE OF AERONAUTICAL ENGINEERING (Autonomous)

ACADEMIC REGULATIONS – BT23

B.Tech. Regular Four-Year Degree Program (for the batches admitted from the academic year 2023 - 2024) & **B.Tech. (Lateral Entry Scheme)** (for the batches admitted from the academic year 2024 - 2025)

For pursuing four year undergraduate Bachelor of Technology (B.Tech) degree program of study in engineering offered by Institute of Aeronautical Engineering under Autonomous status.

A student shall undergo the prescribed courses as given in the program curriculum to obtain his / her degree in major in which he/she is admitted with 160 credits in the entire program of 4 years. Additional 20/18 credits can be acquired for the degree of B.Tech with **Honours or Minor in Engineering**. These additional 20/18 credits will have to be acquired with massive open online courses (MOOCs) / courses offered by the respective department, to tap the zeal and excitement of learning beyond the classrooms. This creates an excellent opportunity for students to acquire the necessary skill set for employability through massive open online courses where the rare expertise of world-famous experts from academics and industry are available.

Separate certificate will be issued in addition to major degree program mentioning that the student has cleared Honours / Minor specialization in respective courses.

1. CHOICE BASED CREDIT SYSTEM

The credit-based semester system provides flexibility in designing program curriculum and assigning credits based on the course content and hours of teaching. The Choice Based Credit System (CBCS) provides a 'cafeteria' type approach in which the students can take courses of their choice, learn at their own pace, undergo additional courses and acquire more than the required credits, and adopt an interdisciplinary approach to learning.

A course defines learning objectives and learning outcomes and comprises lectures / tutorials / laboratory work / field work / project work / seminars / assignments / MOOCs / alternative assessment tools / presentations / self-study etc., or a combination of some of these. Under the CBCS, the requirement for awarding a degree is prescribed in terms of number of credits to be completed by the students.

2. MEDIUM OF INSTRUCTION

The medium of instruction shall be **English** for all courses, examinations, seminar presentations and project work. The program curriculum will comprise courses of study as given in course structure, in accordance with the prescribed syllabi.

3. PROGRAMS OFFERED

Presently, the institute is offering Bachelor of Technology (B.Tech) degree programs in eleven disciplines. The various programs and their two-letter unique codes are given in Table 1.

Table 1: B.Tech Programs offered

S. No	Name of the Program	Title	Code
1	Aeronautical Engineering	AE	07
2	Computer Science and Engineering	CS	05

3	Computer Science and Engineering (AI & ML)	CA	34
4	Computer Science and Engineering (Data Science)	CD	35
5	Computer Science and Engineering (Cyber Security)	CC	36
6	Information Technology	IT	06
7	Electronics and Communication Engineering	EC	04
8	Electrical and Electronics Engineering	EE	02
9	Mechanical Engineering	ME	03
10	Civil Engineering	CE	01

4. SEMESTER STRUCTURE

Each academic year is divided into two semesters, **ODD and EVEN** semester. Both the semesters have regular class work.

- 4.1 Each semester shall be of 21 weeks (Table 2) duration, and this period includes time for course registration, regular class work, examination preparation, and conduction of examinations.
- 4.2 Each semester shall have a minimum of 90 Instructional / working days.
- 4.3 The academic calendar for both Odd and Even semester shown in Table 2 is declared at the beginning of the academic year.

Table 2: Academic Calendar

FIRST SEMESTER (21 weeks)	I Spell Instruction Period	8 weeks	19 weeks
	I Continuous Internal Assessment Examinations (Mid-term)	1 week	
	II Spell Instruction Period	8 weeks	
	II Continuous Internal Assessment Examinations (Mid-term)	1 week	
	Preparation and Practical Examinations	1 week	
	Semester End Examinations		2 weeks
Semester Break and Supplementary Exams			5 weeks
SECOND SEMESTER (21 weeks)	I Spell Instruction Period	8 weeks	19 weeks
	I Continuous Internal Assessment Examinations (Mid-term)	1 week	
	II Spell Instruction Period	8 weeks	
	II Continuous Internal Assessment Examinations (Mid-term)	1 week	
	Preparation and Practical Examinations	1 week	
	Semester End Examinations		2 weeks
Semester Break and Supplementary Exams			5 weeks

- 4.4 Students admitted on transfer from JNTUH affiliated institutes, Universities and other institutes in the courses in which they are required to earn credits so as to be on par with regular students as prescribed by concerned 'Board of Studies'.

5 REGISTRATION / DROPPING / WITHDRAWAL

The academic calendar includes important academic activities to assist the students and the faculty. These include, dates assigned for registration of courses, dropping of courses and withdrawal from courses. This enables the students to be well prepared and take full advantage of the flexibility provided by the credit system.

- 5.1 Each student has to compulsorily register for course work at the beginning of each semester as per the schedule mentioned in the Academic Calendar. It is compulsory for the student to register for

courses in time. The registration will be organized departmentally under the supervision of the Head of the department.

- 5.2. In ABSENTIA, registration will not be permitted under any circumstances.
- 5.3. At the time of registration, students should have cleared all the dues of Institute and Hostel for the previous semesters, paid the prescribed fees for the current semester and not been debarred from the institute for a specified period on disciplinary or any other ground.
- 5.4. In the first two semesters, the prescribed course load per semester is fixed and is mandated to register all courses. Withdrawal / dropping of courses in the first and second semester is not allowed.
- 5.5. In all semesters, the average load is 20 credits / semester, with its minimum and maximum limits being set at 16 and 24 credits. This flexibility enables students (**from IV semester onwards**) to cope-up with the course work considering the academic strength and capability of student.

Note: A course may be offered to the students, only if a minimum of 20 students opt for it.

5.6. Dropping of Courses:

Within one week after the last date of first continuous internal examination, the student may in consultation with his / her faculty mentor / adviser, drop one or more courses without prejudice to the minimum number of credits as specified in clause 5.5. The dropped courses are not recorded in the memorandum of grades. Student must complete the dropped course(s) by registering in the forthcoming semester in order to earn the required credits.

5.7. Withdrawal from Courses:

A student is permitted to withdraw from a course before commencement first continuous internal examination. Such withdrawals will be permitted without prejudice to the minimum number of credits as specified in clause 5.5. A student cannot withdraw a course more than once and withdrawal of reregistered courses is not permitted.

6. CREDIT SYSTEM

The B.Tech Program shall consist of a number of courses and each course shall be assigned with credits. The curriculum shall comprise Program Core Courses (PCC), Program Elective Courses (PEC), Open Elective Courses (OEC), Laboratory Courses, Mandatory Courses (MC), Value Added Courses (VAC), Experiential Engineering Education (ExEEd), Internship and Project work.

Depending on the complexity and volume of the course, the number of contact periods per week will be assigned. Each theory and laboratory course carries credits based on the number of hours / week.

- Contact classes (Theory): 1 credit per lecture hour per week, 1 credit per tutorial hour per week.
- Laboratory hours (Practical): 1 credit for 2 practical hours per week.
- Project work: 1 credit for 2 hours of project work per week.
- Experiential Engineering Education (ExEEd): 1 credit for two per hours.
- Mandatory courses / Value added courses : No credit is awarded.

Credit distribution for courses offered is given in Table 3.

Table 3: Credit distribution

S. No	Course	Hours	Credits
1	Theory courses	2 / 3 / 4	2 / 3 / 4
2	Program elective courses / Open elective courses	3 / 2	3 / 2
3	Laboratory courses	2 / 3 / 4	1 / 1.5 / 2
4	Mandatory course / Value added course	-	0
5	Project Work: Phase - I and II	-	14
6	Full Semester Internship (FSI) Project work	-	14

Major benefits of adopting the credit system are listed below:

- Quantification and uniformity in the listing of courses for all programs at institute, like core, electives and project work.
- Ease of allocation of courses under different heads by using their credits to meet national /international practices in technical education.
- Convenience to specify the minimum / maximum limits of course load and its average per semester in the form of credits to be earned by a student.
- Flexibility in program duration for students by enabling them to pace their course load within minimum/maximum limits based on their preparation and capabilities.
- Wider choice of courses available from any department of the same institute or even from other similar institute, either for credit or for audit.
- Improved facility for students to optimize their learning by availing of transfer of credits earned by them from one College to another.

7. CURRICULAR COMPONENTS

Courses in a curriculum may be of three kinds: **Foundation / Skill, Program core courses, Program elective courses and Open elective courses.**

Foundation / Skill Course:

Foundation courses are the courses based upon the content leads to enhancement of skill and knowledge as well as value based and are aimed at man making education. Skill courses are those areas in which one needs to develop a set of skills to learn anything at all. They are fundamental to learning any course.

Program core courses (PCC):

There may be a program core course in every semester. This is the course which is to be compulsorily studied by a student as a core requirement to complete the requirement of a program in the said discipline of study.

Program elective courses (PEC) / Open elective courses (OEC):

Electives provide breadth of experience in respective branch and application areas. The program elective course(s) is a course which can be chosen from a pool of courses. It may be:

- Supportive to the discipline of study
- Providing an expanded scope
- Enabling an exposure to some other discipline / domain
- Nurturing student's proficiency / skill.

An elective may be program elective, is a discipline centric focusing on those courses which add generic proficiency to the students or may be Open elective course, chosen from unrelated disciplines.

There is list of professional elective courses; students can choose not more than two courses from each track. Overall, students can opt for six professional elective courses which suit their project work in consultation with the faculty advisor / mentor. Nevertheless, one course from each of the three open electives has to be selected. A student may also opt for more elective courses in his/her area of interest.

Every course of the B.Tech program will be placed in one of the eight categories with minimum credits as listed in the Table 4.

Table 4: Category Wise Distribution of Credits

S. No	Category	Breakup of Credits
1	Humanities and Social Sciences (HSMC), including Management.	07
2	Basic Science Courses (BSC) including Mathematics, Physics and Chemistry.	28
3	Engineering Science Courses (ESC), including Workshop, Drawing, ExEEd, Basics of Electrical / Electronics / Mechanical / Computer Engineering.	09

4	Program Core Courses (PCC), relevant to the chosen specialization / branch.	75
5	Program Elective Courses (PEC), relevant to the chosen specialization / branch.	18
6	Open Elective Courses (OEC), from other technical and / or emerging course areas.	09
7	Project work (PROJ) / Full Semester Internship (FSI) Project work	14
8	Mandatory Courses (MC) / Value Added Courses (VAC)	Non-Credit
TOTAL		160

Semester wise course break-up

Following are the **TWO** models of course structure out of which any student shall choose or will be allotted with one model based on their academic performance.

- Full Semester Internship (FSI) Model and
- Non Full Semester Internship (NFSI) Model

A student who secures a minimum CGPA of 7.5 upto IV semester with **no current arrears** and maintains the CGPA of 7.5 till VI semester shall be eligible to opt for FSI. Students can opt full semester internship (FSI) either in VII or VIII semesters. In Non-FSI Model, all the selected students shall carry out the course work and project work as specified in the course catalogue.

8. EVALUATION METHODOLOGY

Total marks for each course shall be based on Continuous Internal Assessment (CIA) and Semester End Examinations (SEE). There shall have a uniform pattern of 40:60 for CIA and SEE of both theory and practical courses. The institute shall conduct multiple continuous internal assessments (CIA) for theory courses. All the performances of a student shall be considered for Continuous Internal Assessment (CIA) marks.

Table 5: Outline for Continuous Internal Assessments (CIA-1 and CIA-2) and SEE:

Activities	CIA-1	CIA-2	SEE	Total Marks
Continuous Internal Examination (CIE)	10 marks	10 marks		20 marks
Definitions and Terminology / Quiz	5 marks	5 marks		10 marks
Tech Talk / Assignment	5 marks	5 marks		10 marks
Semester End Examination (SEE)			60 marks	60 marks
Total	--	--	100 marks	

8.1 Continuous Internal Assessments (CIA-1 and CIA-2)

Assessment is an ongoing process that begins with establishing clear and measurable expected outcomes of student learning, provides students with sufficient opportunities to achieve those outcomes, and concludes with gathering and interpreting evidence to determine how well students' learning matches expectations.

The first component (CIA-1) of assessment is for 10 marks, definitions and terminology / quiz carry 05 marks and 05 marks allotted for Tech talk / Assignments. This assessment and score process should be completed after completing of first 50% of syllabus ($2^{1/2}$) of the course/s and within 45 working days of semester program.

The second component (CIA-2) of assessment is for 10 marks, definitions and terminology / quiz carry 05 marks and 05 marks are allotted for Tech talk / Assignments. This assessment and score process should be completed after completing of remaining 50% of syllabus ($2^{1/2}$) of the course/s and within 45 working days of semester program.

In case of a student who has failed to attend the CIA1 or CIA2 on a scheduled date, shall be deemed that the student has dropped the examination. However, in case a student could not take the test on scheduled date due to genuine reasons, may appeal to the HOD / Principal. The HOD / Principal in consultation with the class in-charge shall decide about the genuineness of the case and decide to conduct Make-Up Examination to such candidate on the date fixed by the Examinations Control Office but before commencement of the concerned semester end examinations.

8.2 Definitions and Terminology / Quiz

Definitions and Terminology: The conduction of definitions and terminology is completely online. The faculty should prepare 30 to 35 questions from each and every module by clearly mentioning the answer. The course handling faculty needs to submit detailed document to the respective department.

Mode of conducting examination: Online through e-Examdesk

Quiz: The faculty should prepare 50 multiple choice questions from each module ($50 * 5 = 250$ questions). Each question will have four choices / options, fill in the blanks etc. The course handling faculty needs to submit detailed document to the respective department.

Mode of conducting examination: Online through e-Examdesk

8.3 Tech Talk / Assignment

Tech Talk: Technical talks cover a wide range of technical concepts and ideas. For conduction of Tech Talk faculty has to submit latest topics from IEEE, CSI, magazines etc.

Assignments: The assignments develop different skills and increase their knowledge base significantly. It provides the evidence for the faculty that the students have achieved the goals. It helps the faculty to evaluate the student's understanding of the course. The output can be judged using sensory perception (observing, reading, tasting etc.). Faculty should prepare 30 to 35 assignment questions from each module and submit the same to respective department.

8.4 Semester End Examination (SEE)

The semester end examinations (SEE), for theory courses, will be conducted for 60 marks. The syllabus for the theory courses is divided into FIVE modules and each module carries equal weightage in terms of marks distribution. The question paper has eight questions; module one and two has one question each, and two questions each from modules three, four and five. There will be no choice for first and second modules. From third module onwards there will be an “either” “or” choice, and the student should answer either of the two questions. Each question carries 12 marks and has two parts (A and B). Part A is a descriptive type question for 6 marks and Part B a critical thinking question / problem solving question for 6 marks. **The duration of semester end examination is 3 hours.**

8.5 Passing Criteria:

To maintain high standards in all aspects of examinations at the institute, the institute shall follow the standards of passing at CIA (CIA-1 and CIA-2) and SEE for each course. However, the student's performance in a course shall be judged by taking into account the results of CIE and SEE individually and also together, as shown below:

- A minimum of 35% of marks to be secured by cumulating marks in CIA-1 and CIA-2 for appearing for a SEE theory examination.
- A minimum of 35% of marks to be scored in SEE for passing a theory course.
- A minimum of 40% of marks in CIA+ SEE for passing a theory course.

8.5 Supplementary examinations

Supplementary examinations for the odd semester shall be conducted with the regular examinations of even semester and vice versa. In case of failure in any course, a student may be permitted to register for the same course when offered.

Advanced supplementary examination will be conducted for VIII semester courses at the end of the program after declaration of results.

8.6 Laboratory Course

Evaluation methodology of laboratory course (CIA)

Each laboratory courses there shall be a CIA during the semester for 40 marks and 60 marks for SEE. The 40 marks for internal evaluation marks are awarded as follows:

1. A write-up on day-to-day experiment in the laboratory (in terms of aim, components / procedure, expected outcome) which shall be evaluated for **10 marks**.
2. **10 marks** for viva-voce (or) tutorial (or) case study (or) application (or) poster presentation of the course concerned.
3. Internal practical examination conducted by the laboratory teacher concerned shall be evaluated for **10 marks**.
4. The remaining **10 marks** are for Laboratory Report/Project and Presentation, which consists of the Design (or) Software / Hardware Model Presentation (or) App Development (or) Prototype Presentation which shall be evaluated after completion of laboratory course and before semester end practical examination.

Evaluation methodology of laboratory course (SEE)

The Semester End Examination shall be conducted by an external examiner and the laboratory handling faculty. The external examiner shall be appointed from the other colleges which will be decided by the Principal.

The Semester End Examination held for 3 hours. Total 60 marks are divided and allocated as shown below:

1. 10 marks for write-up
2. 15 for experiment / program
3. 15 for evaluation of results
4. 10 marks for presentation on another experiment/program in the same laboratory course and
5. 10 marks for viva-voce on concerned laboratory course.

8.7 Mandatory Courses (MC)

These courses are among the compulsory courses will not carry any credits. However, a pass in each such course during the program shall be necessary requirement for the student to qualify for the award of degree. No marks or letter grades shall be allotted for mandatory/non-credit courses. Its result shall be declared as **“Satisfactory” or “Not Satisfactory”** performance.

8.8 Additional Mandatory Courses for lateral entry B.Tech students

In addition to the non-credit mandatory courses for regular B.Tech students, the lateral entry students shall take up the following three non-credit mandatory bridge courses (one in III semester, one in IV semester and one in V semester) as listed in Table 6. The student shall pass the following non-credit mandatory courses for the award of the degree and must clear these bridge courses before advancing to the VII semester of the program.

Table-6: Additional Mandatory Courses for lateral entry students

S. No	Additional mandatory courses for lateral entry students
1	Dip-Mathematics
2	Dip-English Communication Skills
3	Dip-Programming for Problem Solving / Essentials of Problem Solving

8.9 Value Added Courses (VAC)

1. Introduction

Value-Added courses are not part of the curriculum and designed to provide necessary skills to increase the employability quotient and equip the students with essential skills to succeed in life.

Institute offers a wide variety of value-added Courses which shall be conducted after class hours. These courses shall be conducted by experts or in-house staff and help students stand apart from the rest in the job market by adding further value to their resume. These value-added courses will be mostly independent to each type of the fields.

2. Objectives

Objectives of the value-added course are:

- Provide students an understanding of the expectations of industry.
- Improve employability skills of students.
- Bridge the skill gaps and make students industry ready.
- Provide an opportunity to students develop their inter-disciplinary skills.
- Mould students as job providers rather than job seekers.

3. Designing the Courses

- Before designing the syllabus, the feedback from the employers, alumni and industry people will be analyzed and considered to select and design an appropriate course by identifying the gaps and also understand the expectations for current and emerging trends.
- Any new value-added course developed by a department should be placed before the Board of Studies for approval.
- The course offered should not be the same as any course listed in the curriculum of the respective Program / or any other program offered in the institute.
- A unique course code is to be given for each course.

4. Guidelines for conducting value added courses

- Value Added Course is not mandatory to qualify for any program.
- It is a teacher assisted learning course open to all students without any additional fee.
- Classes for VAC will be conducted beyond the regular class work only.
- A student will be permitted to register only one value added course in a semester.

5. Duration and Venue

- The duration of value-added course should not be less than 30 hours.
- The respective Head of the department shall provide class room/s based on the number of students/batches.

6. Procedure for Registration:

The list of value-added courses shall be displayed in the institute website along with the syllabus, objectives and outcomes. A student shall register for a value-added course offered during the semester by submitting the duly filled in registration form. The Head of the department shall segregate the list of students enrolled for the value-added course and submit the details to Dean of academics before the start of course.

7. Attendance

Value added course handling faculty shall be responsible for the maintenance of attendance and assessment who have registered for the course.

- The record shall contain details of the students' attendance, number of classes attended and also Record shall also contain the organisation of lesson plan of the Course Instructor.
- The record shall be submitted to the Head of the department for monitoring the attendance.
- At the end of the semester, the record shall be duly signed by the course coordinator and the Head of the Department and placed in safe custody for any future verification.
- Each student shall have a minimum of 75% attendance in all the courses of the particular semester failing which he or she will not be permitted to write the semester end examination.

8. Passing Requirement and Grading

- The passing requirement for value added courses shall be 40% of the marks prescribed for the course.
- A candidate who has not secured a minimum of 40% of marks in a course (internal and end-term) shall reappear for the course in the next semester/year.
- The grades obtained in value-added courses will not be included for calculating the CGPA.

9. Course Completion

- Students will get a certificate after they have registered for, written the exam and successfully passed.
- The students who have successfully completed the value-added Course shall be issued with a Certificate duly signed by the Authorized signatories.

Note: Apart from the above, students can also register and get the value-added course completion certificate by registering the courses from SWAYAM, e-PG patashala (NPTEL).

8.10 Experiential Engineering Education (ExEED)

Engineering entrepreneurship requires strong technical skills in engineering design and computation with key business skills from marketing to business model generation. Students require sufficient skills to innovate in existing companies or create their own.

This course will be evaluated for a total of 100 marks consisting of 40 marks for internal assessment and 60 marks for semester end Examination. Out of 40 marks of internal assessment, students has to submit Innovative Idea in a team of four members in the given format. The semester end examination for 60 marks shall be conducted internally, students has to present the Innovative Idea and it will be evaluated by internal ExEEd faculty with at least one faculty member as examiner from the industry, both nominated by the Principal from the panel of experts recommended by the Dean-CLET.

Center for Outcome Based Teaching and Learning (OBTL) of the institute design and teach ExL power skills courses, to shape the student's future. All the below mentioned Experiential Engineering Education (ExEED) courses are evaluated for one credit each.

- **ExL – Essential of Innovation:** This course creates platform where students experience a hands-on approach to learning about engineering. It focuses on educating the students about diversified platforms for learning the skills, career development, innovations, entrepreneurship etc. Based on the requirements this course is offered in first or second semester.
- **ExL – Prototype / Model Development:** It covers the application of relevant technologies to create interaction prototypes. Students learn about different kinds of prototyping activities involved in designing low-fidelity and high-fidelity prototypes such as POC models, web pages and mobile interfaces etc. This course introduces key concepts, processes and principles of industry driven digital fabrication in a manufacturing environment. Students will undertake small-scale, team-based project work to create fabricated objects that relate to a local industry, organisation or community need or opportunity.

8.11 Project Work / FSI Project Work

This gives students a platform to experience a research driven career in engineering, while developing a device / systems and publishing in reputed SCI / SCOPUS indexed journals and/or filing an **Intellectual Property** (IPR-Patent/Copyright) to aid communities around the world. Students should work individually as per the guidelines issued by head of the department concerned. The benefits to students of this mode of learning include increased engagement, fostering of critical thinking and greater independence.

The topic should be so selected that the students are enabled to complete the work in the stipulated time with the available resources in the respective laboratories. The scope of the work be handling part of the consultancy work, maintenance of the existing equipment, development of new experiment setup or can be

a prelude to the main project with a specific outcome.

Project report will be evaluated for 100 marks in total. Assessment will be done for 100 marks out of which, the supervisor / guide will evaluate for 40 marks based on the work and presentation / execution of the work. Subdivision for the remaining 60 marks is based on publication, report, presentation, execution and viva-voce. Evaluation shall be done by a committee comprising the supervisor, Head of the department, and an examiner nominated by the Principal.

8.12 Skill enhancement project

Students must submit the skill enhancement project report of the specified course which are included in the course catalogue. If the student has failed to submit the report or not reached upto the mark, needs to re-register the course in next semesters till completion.

8.13 Project work

The student's project activity is spread over in VII semester and in VIII semesters. A student shall carry out the project work under the supervision of a faculty member or in collaboration with an Industry, R&D organization or another academic institution / University where sufficient facilities exist to carry out the project work.

Project work (Phase - I) starts in VII semester as it takes a vital role in campus hiring process. Students shall select project titles from their respective logins uploaded by the supervisors at the middle of the VI semester. Two reviews are conducted by department review committee (DRC). Student must submit a project report summarizing the work done up to design phase/prototype by the end of VII semester. The semester end examination for project work (Phase-I) is evaluated based on the project report submitted and a viva-voce exam for 60 marks by a committee comprising the head of the department, the project supervisor and an external examiner nominated by the Principal.

Project Work (Phase - II) starts in VIII semester, and it shall be evaluated for 100 marks out of which 40 marks towards continuous internal assessment and 60 marks for semester end examination. Two reviews are to be conducted by DRC on the progress of the project for 40 marks. The semester end examination shall be based on the final report submitted and a viva-voce exam for 60 marks by a committee comprising the head of the department, the project supervisor and an external examiner nominated by the Principal.

A minimum of 50% of maximum marks shall be obtained to earn the corresponding credits.

8.14 Full Semester Internship (FSI)

FSI is a full semester internship program carry 14 credits. The FSI shall be opted in VII semester or in VIII semester. During the FSI, student has to spend one full semester in an identified industry / firm / R&D organization or another academic institution/University where sufficient facilities exist to carry out the project work.

Following are the evaluation guidelines:

- Quizzes: 2 times
- Quiz #1 - About the industry profile, weightage: 5%
- Quiz #2 - Technical-project related, weightage: 5%
- Seminars - 2 times (once in six weeks), weightage: 7.5% + 7.5%
- Viva-voce: 2 times (once in six weeks), weightage: 7.5% + 7.5%
- Project Report, weightage: 15%
- Internship Diary, weightage: 5 %
- Final Presentation, weightage: 40%

FSI shall be open to all the branches with a ceiling of maximum 10% distributed in both semesters. The selection procedure is:

- Choice of the students.
- CGPA (> 7.5) upto IV semester having no credit arrears.
- Competency Mapping / Allotment.

It is recommended that the FSI Project work leads to a research publication in a reputed Journal / Conference or the filing of patent / design with the patent office, or, the start-up initiative with a sustainable and viable business model accepted by the incubation center of the institute together with the formal registration of the startup.

8.15 Field projects / Internship Academic attachment:

The Field Projects (FP) / Internships are mandatory for the students admitted from the academic year 2023 -24 onwards. It is spreaded over from II semester to VI semester.

Field Project: Field project (FP) integrates theory and practice by providing students with an opportunity to work on real-world challenges. It can be used to learn about the functioning and manufacturing procedures of a factory. Besides this, student can also learn about the geographical factors of the region for the specific products /equipment.

Internship is an integral part of the academic curriculum, it is a learning activity in which a student fortifies and deepens his/her theoretical knowledge and skills attained in the classrooms by integrating with practical activities. It offers the students an opportunity to gain hands-on industrial or organizational exposure; to integrate the knowledge and skills acquired through the coursework; interact with professionals and other interns; and to improve their presentation, writing, and communication skills. Internship often acts as a gateway for final placement for many students.

Table 7: Possibility of availing opportunities during semester breaks.

S.No	Schedule	Duration	Type
1	At the end of II semester / Before commencement of III semester	2 Weeks	Field Project
2	At the end of IV semester / Before commencement of V semester	2 Weeks	Internship
3	At the end of VI semester / Before commencement of VII semester	2 Weeks	Internship

Evaluation Methodology of Field Project / Internships:

The evaluation of the field project / field practicum / Internships will be done before commencement of subsequent semester specified in Table: 7. The students have to submit a detailed report of field project / field practicum / Internships through online portal and also carry hard copy of report with geo-tagged photographs. The committee will evaluate by enclosing their comments like **satisfactory or not satisfactory**. If students get not satisfactory results, reports need to be re-submit in the respective department once again for evaluation.

8.16 Plagiarism index for Project report:

All project reports shall go through the plagiarism check and the plagiarism index has to be less than 20%. Project reports with plagiarism more than 20% and less than 60% shall be asked for resubmission within a stipulated period of six months. Project reports with plagiarism more than 60% shall be rejected.

9. ATTENDANCE REQUIREMENTS AND DETENTION POLICY

- 9.1 A student shall be eligible to appear for the semester end examinations, if the student acquires a minimum of 75% of attendance in aggregate of all the courses (including attendance in mandatory courses like Environmental Science, Constitution of India, and Gender Sensitization etc..) for that semester. **Two periods** of attendance for each theory course shall be considered, if the student appears for the mid-term examination of that course.
- 9.2 Shortage of attendance in aggregate upto 10% (65% and above, and below 75%) in each semester may be condoned by the college academic committee on genuine and valid grounds, based on the student's representation with supporting evidence.
- 9.3 A stipulated fee shall be payable for condoning of shortage of attendance.
- 9.4 Shortage of attendance below 65% in aggregate shall in NO case be condoned.
- 9.5 Students whose shortage of attendance is not condoned in any semester are not eligible to take their semester end examinations of that semester. They get detained and their registration for that semester shall stand cancelled, including all academic credentials (internal marks etc.) of that semester. They will not be promoted to the next semester. They may seek re-registration for all those courses

registered in that semester in which the student is detained, by seeking re-admission into that semester as and when offered; if there are any program electives and / or open electives, the same may also be re- registered if offered. However, if those electives are not offered in later semesters, then alternate electives may be chosen from the **same** set of elective courses offered under that category.

- 9.6 A student fulfilling the attendance requirement in the present semester shall not be eligible for readmission into the same class.
- 9.7 A student detained in a semester due to shortage of attendance may be re-admitted in the same semester in the next academic year for fulfillment of academic requirements. The academic regulations under which a student has been re-admitted shall be applicable. Further, no grade allotments or SGPA/ CGPA calculations will be done for the entire semester in which the student has been detained.
- 9.8 A student detained due to lack of credits, shall be promoted to the next academic year only after acquiring the required number of academic credits. The academic regulations under which the student has been readmitted shall be applicable to him.

10. CONDUCT OF SEMESTER END EXAMINATIONS AND EVALUATION

- 10.1 Semester end examination shall be conducted by the Controller of Examinations (COE) by inviting question papers from the external examiners.
- 10.2 COE shall invite 3 - 9 internal / external examiners to evaluate all the semester end examination answer books on a prescribed date(s). Practical laboratory examinations are conducted involving external examiners.
- 10.3 Examinations control office shall consolidate the marks awarded by examiner/s and award the grades.

11. SCHEME FOR THE AWARD OF GRADE

- 11.1 A student shall be deemed to have satisfied the minimum academic requirements and earn the credits for each theory course, if s/he secures,
 - a) Not less than 35% marks for each theory course in the semester end examination, and
 - b) A minimum of 40% marks for each theory course considering Continuous Internal Assessment (CIA) and Semester End Examination (SEE).
- 11.2 A student shall be deemed to have satisfied the minimum academic requirements and earn the credits for each Laboratory / Project work / FSI Project work, if s/he secures,
 - a) Not less than 40% marks for each Laboratory / Project work / FSI project work course in the semester end examination,
 - b) A minimum of 40% marks for each Laboratory / Project work / FSI project work course considering both internal and semester end examination.
- 11.3 If a candidate fails to secure a pass in a particular course, it is mandatory that s/he shall register and reappear for the examination in that course during the next semester when examination is conducted in that course. It is mandatory that s/he should continue to register and reappear for the examination till s/he secures a pass.
- 11.4 A student shall be declared successful or 'passed' in a semester, if he secures a Grade Point ≥ 5 ('C' grade or above) in every course in that semester (i.e. when the student gets an SGPA ≥ 5.0 at the end of that particular semester); and he shall be declared successful or 'passed' in the entire under graduate programme, only when gets a CGPA ≥ 5.0 for the award of the degree as required.

12. LETTER GRADES AND GRADE POINTS

- 12.1 Performances of students in each course are expressed in terms of marks as well as in Letter Grades based on absolute grading system. The UGC recommends a 10-point grading system with the following letter grades as given in the Table 8.

Table-8: Grade Points Scale (Absolute Grading)

% of Marks Secured in a Course (Class Intervals)	Letter Grade	Grade Point
Greater than or equal to 90%	O (Outstanding)	10
80 and less than 90%	A+ (Excellent)	9
70 and less than 80%	A (Very Good)	8
60 and less than 70%	B+ (Good)	7
50 and less than 60%	B (Average)	6
40 and less than 50%	C (Pass)	5
Below 40%	F (Fail)	0
Absent	AB (Absent)	0

- 12.2 A student is deemed to have passed and acquired to correspondent credits in particular course if s/he obtains any one of the following grades: “O”, “A+”, “A”, “B+”, “B”, “C”.
- 12.3 A student obtaining Grade F shall be considered Failed and will be required to reappear in the examination.
- 12.4 For non credit courses, ‘PP’ or “NP” is indicated instead of the letter grade and this will not be counted for the computation of SGPA / CGPA.
- 12.5 At the end of each semester, the institute issues grade sheet indicating the SGPA and CGPA of the student. However, grade sheet will not be issued to the student if s/he has any outstanding dues.

Table 09: Percentage Equivalence of Grade Points (for a 10 – Point Scale)

Grade Point	Percentage of Marks / Class
5.5	50
6.0	55
6.5	60
7.0	65
7.5	70
8.0	75

Note:

- (1) The following Formula for Conversion of CGPA to percentage of marks to be used only after a student has successfully completed the program:

$$\text{Percentage of Marks} = (\text{CGPA} - 0.5) \times 10$$

- (2) Class designation:

≥75% (First Class with Distinction),
 ≥ 60% and <75 % (First Class),
 <60 % and ≥50% (Second Class),
 ≥40% and <50% (Pass Class).

- (3) The SGPA will be computed and printed on the Memorandum of Grades only if the candidate passes in all the courses offered and gets minimum C grade in all the courses.
- (4) CGPA is calculated only when the candidate passes in all the courses offered in all the semesters.

13. COMPUTATION OF SGPA AND CGPA

The UGC recommends to compute the Semester Grade Point Average (SGPA) and Cumulative Grade Point Average (CGPA). The credit points earned by a student are used for calculating the Semester Grade Point Average (SGPA) and the Cumulative Grade Point Average (CGPA), both of which are important

performance indices of the student. SGPA is equal to the sum of all the total points earned by the student in a given semester divided by the number of credits registered by the student in that semester. CGPA gives the sum of all the total points earned in all the previous semesters and the current semester divided by the number of credits registered in all these semesters. Thus,

$$SGPA = \sum_{i=1}^n (C_i G_i) / \sum_{i=1}^n C_i$$

Where, C_i is the number of credits of the i^{th} course and G_i is the grade point scored by the student in the i^{th} course and n represent the number of courses in which a student is registered in the concerned semester.

$$CGPA = \sum_{j=1}^m (C_j S_j) / \sum_{j=1}^m C_j$$

Where, S_j is the SGPA of the j^{th} semester and C_j is the total number of credits upto the semester and m represent the number of semesters completed in which a student registered upto the semester.

The SGPA and CGPA shall be rounded off to 2 decimal points and reported in the transcripts.

14.0 ILLUSTRATION OF COMPUTATION OF SGPA AND CGPA

14.1 Illustration for SGPA

Course Name	Course Credits	Grade letter	Grade point	Credit Point (Credit x Grade)
Course 1	4	A	8	4 x 8 = 32
Course 2	4	O	10	4 x 10 = 40
Course 3	4	C	5	4 x 5 = 20
Course 4	3	B	6	3 x 6 = 18
Course 5	3	A+	9	3 x 9 = 27
Course 6	3	C	5	3 x 5 = 15
	21			152

Thus, $SGPA = 152 / 21 = 7.24$

14.2 Illustration for calculation of CGPA upto 3rd semester

Semester	Course Title	Credits Allotted	Letter Grade Secured	Corresponding Grade Point (GP)	Credit Points (CP)
I	Course 1	3	A	8	24
I	Course 2	3	O	10	30
I	Course 3	3	B	6	18
I	Course 4	4	A	8	32
I	Course 5	3	A+	9	27
I	Course 6	4	C	5	20
II	Course 7	4	B	6	24
II	Course 8	4	A	8	32
II	Course 9	3	C	5	15
II	Course 10	3	O	10	30
II	Course 11	3	B+	7	21
II	Course 12	4	B	6	24
II	Course 13	4	A	8	32
II	Course 14	3	O	10	30
III	Course 15	2	A	8	16
III	Course 16	1	C	5	5

III	Course 17	4	O	10	40
III	Course 18	3	B+	7	21
III	Course 19	4	B	6	24
III	Course 20	4	A	8	32
III	Course 21	3	B+	7	21
	Total Credits	69		Total Credits	518

Thus, CGPA = 518 / 69 = 7.51

- The calculation process of CGPA illustrated above will be followed for each subsequent semester until 8th semester. The CGPA obtained at the end of 8th semester will become the final CGPA secured for entire B.Tech. program.
- For merit ranking or comparison purposes or any other listing, only the rounded off values of the CGPAs will be used.
- SGPA and CGPA of a semester will be mentioned in the semester Memorandum of Grades if all courses of that semester are passed in first attempt, otherwise the SGPA and CGPA shall be mentioned only on the Memorandum of Grades in which sitting s/he passed his last exam in that semester. However, mandatory courses will not be taken into consideration.

15. REVIEW OF SEE THEORY ANSWER BOOKS

If the examinee is not satisfied with the marks awarded, s/he may apply for revaluation of answer book in prescribed format online within three (3) working days from the date of declaration of result of the examination or issue of the statement of marks, whichever is earlier. The revaluation facility shall be for theory papers only. The revaluation of answer book shall not be permitted in respect of the marks awarded to the scripts of practical examination / project work (including theory part) and in viva voce / oral / comprehensive examinations.

The re-evaluation will be done by a second independent examiner. The result after re-evaluation shall be as follows:

1. The revaluation marks are considered only if the difference between the original award and award on reevaluation is more than equal to 15% of 60 marks (09 marks).
2. If the difference between the original award and the award on reevaluation is more than 20% (12 marks), a third evaluator is to be appointed and the average of two nearest awards (in the range of 15%) shall be considered.

16. PROMOTION POLICIES

The following academic requirements have to be satisfied in addition to the attendance requirements mentioned in item no. 9.

16.1 For students admitted into B.Tech (Regular) program

- 16.1.1 A student will not be promoted from II semester to III semester unless s/he fulfills the academic requirement of securing 50% of the total credits (rounded to the next lowest integer) from I and II semester examinations, whether the candidate takes the examination(s) or not.
- 16.1.2 A student will not be promoted from IV semester to V semester unless s/he fulfills the academic requirement of securing 60% of the total credits (rounded to the next lowest integer) upto III semester **or** 60% of the total credits (rounded to the next lowest integer) up to IV semester, from all the examinations, whether the candidate takes the examination(s) or not.
- 16.1.3 A student shall be promoted from VI semester to VII semester only if s/he fulfills the academic requirements of securing 60% of the total credits (rounded to the next lowest integer) up to V semester **or** 60% of the total credits (rounded to the next lowest integer) up to VI semester from all the examinations, whether the candidate takes the examination(s) or not.
- 16.1.4 A student shall register for all the 160 credits and earn all the 160 credits. Marks obtained in all the 160 credits shall be considered for the award of the Grade.

16.2 For students admitted into B.Tech (Lateral entry students)

- 16.2.1 A student will not be promoted from IV semester to V semester unless s/he fulfills the academic requirement of securing 60% of the total credits (rounded to the next lowest integer) up to IV semester, from all the examinations, whether the candidate takes the examination(s) or not.
- 16.2.2 A student shall be promoted from VI semester to VII semester only if s/he fulfills the academic requirements of securing 60% of the total credits (rounded to the next lowest integer) up to V semester **or** 60% of the total credits (rounded to the next lowest integer) up to VI semester from all the examinations, whether the candidate takes the examination(s) or not.
- 16.2.3 A student shall register for all the 120 credits and earn all the 120 credits. Marks obtained in all the 120 credits shall be considered for the award of the Grade.

17. GRADUATION REQUIREMENTS

The following academic requirements shall be met for the award of the B.Tech degree.

- 17.1 Student shall register and acquire minimum attendance in all courses and secure 160 credits (with minimum CGPA of 5.0), for regular program and 120 credits (with minimum CGPA of 5.0), for lateral entry program. **There is NO exemption of credits in any case.**
- 17.2 A student of a regular program, who fails to earn 160 credits within **eight consecutive academic years** from the year of his/her admission with a minimum CGPA of 5.0, shall forfeit his/her degree and his/her admission stands cancelled.
- 17.3 A student of a lateral entry program who fails to earn 120 credits within **six consecutive academic years** from the year of his/her admission with a minimum CGPA of 5.0, shall forfeit his/her degree and his/her admission stands cancelled.

18. AWARD OF DEGREE

18.1 Classification of degree will be as follows:

CGPA > 8.0	CGPA \geq 7.0 and < 8.0	CGPA \geq 6.0 and < 7.0	CGPA \geq 5.0 and < 6.0	CGPA < 5.0
First Class with Distinction	First Class	Second Class	Pass Class	Fail

- 18.2 A student with final CGPA (at the end of the under graduate programme) >8.0 , and fulfilling the following conditions - shall be placed in '**first class with distinction**'. However,
- Should have passed all the courses in '**first appearance**' within the first 4 academic years (or 8 sequential semesters) from the date of commencement of first semester.
 - Should have secured a CGPA >8.0 , at the end of each of the 8 sequential semesters, starting from first semester onwards.
 - Should not have been detained or prevented from writing the semester end examinations in any semester due to shortage of attendance or any other reason.
 - A student not fulfilling any of the above conditions with final CGPA >8.0 shall be placed in '**first class**'.
- 18.3 Students with final CGPA (at the end of the B.Tech program) ≥ 7.0 but <8.0 shall be placed in '**first class**'.
- 18.4 Students with final CGPA (at the end of the B.Tech program) ≥ 6.0 but <7.0 , shall be placed in '**second class**'.
- 18.5 All other students who qualify for the award of the degree (as per item 18), with final CGPA (at the end of the B.Tech program) ≥ 5.0 but <6.0 , shall be placed in '**pass class**'.
- 18.6 A student with final CGPA (at the end of the B.Tech program) <5.0 will not be eligible for the award of the degree.
- 18.7 Students fulfilling the conditions listed under item 18.2 alone will be eligible for award of '**Gold Medal**'.

All the candidates who register for the semester end examination will be issued a memorandum of grades sheet by the institute. Apart from the semester wise memorandum of grades sheet, the institute will issue the provisional certificate and consolidated grades memorandum course to the fulfillment of all the academic requirements.

19. B.TECH WITH HONOURS OR MINOR IN ENGINEERING

Students acquiring 160 credits are eligible to get B.Tech degree in Engineering. A student will be eligible to get B.Tech degree with Honours or Minor in Engineering, if s/he completes an additional 20/18 credits (3/4 credits per course). These could be acquired through MOOCs from SWAYAM / NPTEL only. The list for MOOCs will be a dynamic one, as new courses are added from time to time. Few essential skill sets required for employability are also identified year wise. Students interested in doing MOOC courses shall register the course title at their department office at the start of the semester against the courses that are announced by the department. Any expense incurred for the MOOC course / summer program should be met by the students.

Students having no credit arrears and a CGPA of 7.5 or above at the end of the fourth semester are eligible to register for B.Tech (Honours / Minor). After registering for the B.Tech (Honours / Minor) program, if a student fails in any course, s/he will not be eligible for B.Tech (Honours / Minor).

Every department should develop and submit a Honours / Minor – courses list of 5 - 6 theory courses, laboratory and project work.

Honours Certificate for Vertical in his/her OWN Branch for Research orientation; Minor in any other branch for Improving Employability.

Honours will be reflected in the degree certificate as “B.Tech (Honours) in XYZ Engineering”. Similarly, Minor as “B.Tech in XYZ Engineering with Minor in ABC”.

19.1. B.Tech with Honours

The key objectives of offering B. Tech. with Honors program are:

- To expand the domain knowledge of the students laterally and vertically.
- To increase the employability of undergraduate students with expanded knowledge in one of the core engineering disciplines.
- To provide an opportunity to students to pursue their higher studies in wider range of specializations.

Academic Regulations for B. Tech. Honours degree

1. The weekly instruction hours, internal and external evaluation and award of grades are on par with regular 4-Years B. Tech. program.
2. For B. Tech with Honors program, a student needs to earn additional 20 credits (over and above the required 160 credits for B. Tech degree). All these 20 credits required to be attained for B.Tech Honors degree credits are distributed from V semester to VII semester.
3. After registering for the Honors program, if a student is unable to pass all courses in first attempt and earn the required 20 credits, he/she shall not be awarded Honours degree. However, if the student earns all the required 160 credits of B. Tech., he/she will be awarded only B. Tech degree in the concerned branch.
4. There is no transfer of credits from courses of Honors program to regular B. Tech. degree course & vice versa.
5. These **20 credits** are to be earned from the additional courses offered by the host department in the institute or from closely related departments in the institute as well as from the MOOCS platform (NPTEL only).
6. The choice to opt/take the Honors program is purely on the choice of the students.
7. The student shall be given a choice of withdrawing all the courses registered and/or the credits earned for Honours program at any time; and in that case the student will be awarded only B.Tech. degree in the concerned branch on earning the required credits of 160.
8. The students of every branch can choose Honours program in their respective branches if they are eligible for the Honors program. A student who chooses an **Honors program is not eligible to choose a Minor program and vice-versa.**

9. A student can graduate with Honors if he/she fulfils the requirements for his/her regular B.Tech. program as well as fulfills the requirements for Honours program.
10. The institute shall maintain a record of students registered and pursuing their Honors programs branch-wise.
11. The department shall prepare the time-tables for each Honors program offered at their respective departments without any overlap/clash with other courses of study in the respective semesters.

Eligibility conditions of the students for the B.Tech Honors degree

- a) A student can opt for B.Tech. degree with Honors, if she/he passed all courses in first attempt in all the semesters till the results announced and maintaining **7.5 or more CGPA**.
- b) If a student fails in any registered course of either B.Tech or Honours in any semester of four years program, he/she will not be eligible for obtaining Honors degree. He will be eligible for only
- c) Prior approval of mentor and Head of the Department for the enrolment into Honors program, before commencement of V Semester, is mandatory.
- d) If more than **30% of the students** in a branch fulfill the eligibility criteria (as stated above), the number of students given eligibility should be limited to 30%. The criteria to be followed for choosing 30% candidates in a branch may be the CGPA secured by the students till **III** semester.
- e) Successful completion of **20 credits** earmarked for Honours program with at least 7.5 CGPA along with successful completion of 160 credits earmarked for regular B. Tech. Program with at least 7.5 CGPA and passing all courses in first attempt gives the eligibility for the award of B.Tech (Honors) degree.
- f) For CGPA calculation of B.Tech. course, the **20 credits** of Honours program will not be considered.

Following are the details of such Honors which include some of the most interesting areas in the profession today:

S. No	Department	Honors scheme
1	Aeronautical Engineering	Aerospace Engineering / Space Science etc.
2	Computer Science and Engineering / Information Technology	Big data and Analytics / Cyber Physical Systems, Information Security / Cognitive Science / Artificial Intelligence/ Machine Learning / Data Science / Internet of Things (IoT) / Cyber Security etc.
3	Electronics and Communication Engineering	Digital Communication / Signal Processing / Communication Networks / VLSI Design / Embedded Systems etc.
4	Electrical and Electronics Engineering	Renewable Energy systems / Energy and Sustainability / IoT Applications in Green Energy Systems etc.
5	Mechanical Engineering	Industrial Automation and Robotics / Manufacturing Sciences and Computation Techniques etc.
6	Civil Engineering	Structural Engineering / Environmental Engineering etc.

19.2 B.Tech with Minor in Engineering

The key objectives of offering B.Tech with Minor program are:

- To expand the domain knowledge of the students in one of the other branches of engineering.
- To increase the employability of undergraduate students keeping in view of better opportunity in interdisciplinary areas of engineering & technology.
- To provide an opportunity to students to pursue their higher studies in the inter-disciplinary areas in addition to their own branch of study.
- To offer the knowledge in the areas which are identified as emerging technologies/thrust areas of Engineering.

Advantages of Minor in Engineering

The minors mentioned above are having lots of advantages and a few are listed below:

- To enable students to pursue allied academic interest in contemporary areas.
- To provide an academic mechanism for fulfilling multidisciplinary demands of industries.
- To provide effective yet flexible options for students to achieve basic to intermediate level competence in the Minor area.
- Provides an opportunity to students to become entrepreneurs and leaders by taking business/management minor.
- Combination in the diverse fields of engineering e.g., CSE (Major) + Electronics (Minor) combination increases placement prospects in chip designing companies.
- Provides an opportunity for applicants to pursue higher studies in an inter-disciplinary field of study.
- To increase the overall scope of the undergraduate degrees.

Academic Regulations for B.Tech Degree with Minor programs

1. The weekly instruction hours, internal & external evaluation and award of grades are on par with regular 4-Years B. Tech. program.
2. For B. Tech. with Minor, a student needs to earn additional 18 credits (over and above the required 160 credits for B. Tech degree). The courses are offered from V semester to VII semester only, to obtain minor degree students required to obtain 18 credits.
3. After registering for the Minor program, if a student is unable to earn all the required 18 credits in a specified duration (twice the duration of the course), he/she shall not be awarded Minor degree. However, if the student earns all the required 160 credits of B.Tech, he/she will be awarded only B. Tech degree in the concerned branch.
4. There is no transfer of credits from Minor program courses to regular B. Tech. degree course & vice versa.
5. These 18 credits are to be earned from the additional courses offered by the host department in the institute as well as from the MOOCs platform.
6. For the course selected under MOOCs platform (NPTEL) following guidelines may be followed:
 - a) Prior to registration of MOOCs courses, formal approval of the courses, by the institute is essential, before the issue of approval considers the parameters like the institute / agency which is offering the course, syllabus, credits, duration of the program and mode of evaluation etc.
 - b) Minimum credits for MOOCs course must be equal to or more than the credits specified in the Minor course structure provided by the institute.
 - c) Only Pass-grade / marks or above shall be considered for inclusion of grades in minor grade memo.
 - d) Any expenses incurred for the MOOCs courses are to be met by the students only.
7. The choice to opt / take a Minor program is purely on the choice of the students.
8. The student shall be given a choice of withdrawing all the courses registered and / or the credits earned for Minor program at any time; and in that case the student will be awarded only B. Tech. degree in the concerned branch on earning the required credits of 160.
9. The student can choose only one Minor program along with his / her basic engineering degree. A student who chooses an **Honors program is not eligible to choose a Minor program and vice-versa.**
10. The institute shall maintain a record of students registered and pursuing their Minor programs, minor program-wise and parent branch-wise.
11. The institute / department shall prepare the time-tables for each Minor course offered at their respective institutes without any overlap/clash with other courses of study in the respective semesters.

Eligibility conditions for the student to register for Minor course

- a) A student can opt for B.Tech. degree with Minor program if she/he has no active backlogs till III semester at the time of entering into III year I semester.
- b) Prior approval of mentor and Head of the Department for the enrolment into Minor program, before commencement of III year I Semester (V Semester), is mandatory.
- c) If more than 50% of the students in a branch fulfill the eligibility criteria (as stated above), the number of students given eligibility should be limited to 50%.

20.0 TEMPORARY BREAK OF STUDY FROM THE PROGRAM

- 20.1 A candidate is normally not permitted to take a break from the study. However, if a candidate intends to temporarily discontinue the program in the middle for valid reasons (such as accident or hospitalization due to prolonged ill health) and to rejoin the program in a later respective semester, s/he shall seek the approval from the Principal in advance. Such application shall be submitted before the last date for payment of examination fee of the semester and forwarded through the Head of the Department stating the reasons for such withdrawal together with supporting documents and endorsement of his / her parent / guardian.
- 20.2 The institute shall examine such an application and if it finds the case to be genuine, it may permit the student to temporarily withdraw from the program. Such permission is accorded only to those who do not have any outstanding dues / demand at the College / University level including tuition fees, any other fees, library materials etc.
- 20.3 The candidate has to rejoin the program after the break from the commencement of the respective semester as and when it is offered.
- 20.4 The total period for completion of the program reckoned from the commencement of the semester to which the candidate was first admitted shall not exceed the maximum period specified in clause 17. The maximum period includes the break period.
- 20.5 If any candidate is detained for any reason, the period of detention shall not be considered as 'Break of Study'.

21. TERMINATION FROM THE PROGRAM

The admission of a student to the program may be terminated and the student is asked to leave the institute in the following circumstances:

- a. The student fails to satisfy the requirements of the program within the maximum period stipulated for that program.
- b. A student shall not be permitted to study any semester more than three times during the entire program of study.
- c. The student fails to satisfy the norms of discipline specified by the institute from time to time.

22. WITH-HOLDING OF RESULTS

If the candidate has not paid any dues to the institute / if any case of indiscipline / malpractice is pending against him, the results and the degree of the candidate will be withheld.

23. GRADUATION DAY

The institute shall have its own annual Graduation Day for the award of degrees to the students completing the prescribed academic requirements in each case, in consultation with the University and by following the provisions in the Statute. The college shall institute prizes and medals to meritorious students and award them annually at the Graduation Day. This will greatly encourage the students to strive for excellence in their academic work.

24. DISCIPLINE

Every student is required to observe discipline and decorum both inside and outside the institute and are expected not to indulge in any activity which will tend to bring down the honour of the institute. If a student indulges in malpractice in any of the theory / practical examination, continuous assessment

examinations, he/she shall be liable for punitive action as prescribed by the institute from time to time.

25. GRIEVANCE REDRESSAL COMMITTEE

The institute shall form a Grievance Redressal Committee for each course in each department with the Course Teacher and the HOD as the members. The committee shall solve all grievances related to the course under consideration.

26. TRANSITORY REGULATIONS

A candidate, who is detained or has discontinued a semester, on readmission shall be required to do all the courses in the curriculum prescribed for the batch of students in which the student joins subsequently. However, exemption will be given to those candidates who have already passed such courses in the earlier semester(s) he was originally admitted into and substitute courses were offered in place of them as decided by the Board of Studies. However, the decision of the Board of Studies will be final.

a) Transfer candidates (from non-autonomous college affiliated to JNTUH):

A student who is following JNTUH curriculum, transferred from other college to this institute in third semester or subsequent semesters shall join with the autonomous batch in the appropriate semester. Such candidates shall be required to pass in all the courses in the program prescribed by the Board of Studies concerned for that batch of students from that semester onwards to be eligible for the award of degree. However, exemption will be given in the courses of the semester(s) of the batch which he had passed earlier and substitute courses are offered in their place as decided by the Board of Studies. The student has to clear all his backlog courses up to previous semester by appearing for the supplementary examinations conducted by JNTUH for the award of degree. The total number of credits to be secured for the award of the degree will be the sum of the credits up to the previous semester under JNTUH regulations and the credits prescribed for the semester in which a candidate joined after transfer and subsequent semesters under the autonomous status. The class will be awarded based on the academic performance of a student in the autonomous pattern.

b) Transfer candidates (from an autonomous college affiliated to JNTUH):

A student who has secured the required credits up to previous semester as per the regulations of other autonomous institutions shall also be permitted to be transferred to this institute. A student who is transferred from the other autonomous colleges to this institute in third semester or subsequent semesters shall join with the autonomous batch in the appropriate semester. Such candidates shall be required to pass in all the courses in the program prescribed by the Board of Studies concerned for that batch of students from that semester onwards to be eligible for the award of degree. However, exemption will be given in the courses of the semester(s) of the batch which he had passed earlier and substitute courses are offered in their place as decided by the Board of Studies. The total number of credits to be secured for the award of the degree will be the sum of the credits up to previous semester as per the regulations of the college from which he is transferred and the credits prescribed for the semester in which a candidate joined after transfer and subsequent semesters under the autonomous status. The class will be awarded based on the academic performance of a student in the autonomous pattern.

c) Readmission from R18 / UG20 to BT23 regulations

A student took admission in BT23 Regulations, detained due to lack of required number of credits or shortage of attendance at the end of any semester is permitted to take re-admission at appropriate level under any regulations prevailing in the institute course to the following rules and regulations.

1. Student shall pass all the courses in the earlier scheme of regulations. However, in case of having backlog courses, they shall be cleared by appearing for supplementary examinations conducted under earlier regulations from time to time.
2. After readmission, the student is required to study the courses as prescribed in the new regulations for the re-admitted program at that level and thereafter.
3. If the student has already passed any course(s) of readmitted program in the earlier regulation / semester of study, such courses are exempted.

4. The courses that are not done in the earlier regulations / semester as compared need to be cleared after readmission by appearing for the examinations conducted time to time under the new regulations.
5. In general, after transition, course composition and number of credits / semester shall be balanced between old and new regulations on case to case basis.
6. In case, the students who do not have option of acquiring required credits with the existing courses offered as per the new curriculum under autonomy, credit balance can be achieved by clearing the additional courses offered. The additional courses that are offered can be of theory or laboratory courses.

27. REVISION OF REGULATIONS AND CURRICULUM

The Institute from time to time may revise, amend or change the regulations, scheme of examinations and syllabi if found necessary and on approval by the Academic Council and the Governing Body shall be binding on the students, faculty, staff, all authorities of the Institute and others concerned.

FREQUENTLY ASKED QUESTIONS AND ANSWERS ABOUT AUTONOMY

1. Who grants Autonomy? UGC, Govt., AICTE or University

In case of Colleges affiliated to a university and where statutes for grant of autonomy are ready, it is the respective University that finally grants autonomy but only after concurrence from the respective state Government as well as UGC. The State Government has its own powers to grant autonomy directly to Govt. and Govt. aided Colleges.

2 Shall IARE award its own Degrees?

No. Degree will be awarded by Jawaharlal Nehru Technological University, Hyderabad with a mention of the name IARE on the Degree Certificate.

3 What is the difference between a Deemed University and an Autonomy College?

A Deemed University is fully autonomous to the extent of awarding its own Degree. A Deemed University is usually a Non-Affiliating version of a University and has similar responsibilities like any University. An Autonomous College enjoys Academic Autonomy alone. The University to which an autonomous college is affiliated will have checks on the performance of the autonomous college.

4 How will the Foreign Universities or other stake – holders know that we are an Autonomous College?

Autonomous status, once declared, shall be accepted by all the stake holders. The Govt. of Telangana mentions autonomous status during the First Year admission procedure. Foreign Universities and Indian Industries will know our status through our website.

5 What is the change of Status for Students and Teachers if we become Autonomous?

An autonomous college carries a prestigious image. Autonomy is actually earned out of our continued past efforts on academic performances, our capability of self- governance and the kind of quality education we offer.

6 Who will check whether the academic standard is maintained / improved after Autonomy? How will it be checked?

There is a built in mechanism in the autonomous working for this purpose. An Internal Committee called Academic Program Evaluation Committee, which will keep a watch on the academics and keep its reports and recommendations every year. In addition the highest academic council also supervises the academic matters. The standards of our question papers, the regularity of academic calendar, attendance of students, speed and transparency of result declaration and such other parameters are involved in this process.

7 Will the students of IARE as an Autonomous College qualify for University Medals and Prizes for academic excellence?

No. IARE has instituted its own awards, medals, etc. for the academic performance of the students. However for all other events like sports, cultural on co-curricular organized by the University the students shall qualify.

8 Can IARE have its own Convocation?

No. Since the University awards the degree the convocation will be that of the University, but there will be Graduation Day at IARE.

9 Can IARE give a provisional degree certificate?

Since the examinations are conducted by IARE and the results are also declared by IARE, the college sends a list of successful candidates with their final Grades and Grade Point Averages including CGPA to the affiliating university. Therefore with the prior permission of the university the institute will be entitled to give the provisional certificate.

10 Will Academic Autonomy make a positive impact on the Placements or Employability?

Certainly. The number of students qualifying for placement interviews is expected to improve, due to

rigorous and repetitive classroom teaching and continuous assessment. Also the autonomous status is more responsive to the needs of the industry. As a result therefore, there will be a lot of scope for industry oriented skill development built-in into the system. The graduates from an autonomous college will therefore represent better employability.

11 What is the proportion of Internal and External Assessment as an Autonomous College?

Presently, it is 60% external and 40% internal. As the autonomy matures the internal assessment component shall be increased at the cost of external assessment.

12 Is it possible to have complete Internal Assessment for Theory or Practicals?

Yes indeed. We define our own system. We have the freedom to keep the proportion of external and internal assessment component to choose.

13 Why Credit based Grade System?

The credit based grade system is an accepted standard of academic performance the world over in all Universities. The acceptability of our graduates in the world market shall improve.

14 What exactly is a Credit based Grade System?

The credit based grade system defines a much better statistical way of judging the academic performance. One Lecture Hour per week of Teaching Learning process is assigned One Credit. One hour of laboratory work is assigned half credit. Letter Grades like A, B,C,D, etc. are assigned for a Range of Marks. (e.g. 91% and above is A+, 80 to 90 % could be A etc.) in Absolute Grading System while grades are awarded by statistical analysis in relative grading system. We thus dispense with sharp numerical boundaries. Secondly, the grades are associated with defined Grade Points in the scale of 1 to 10. Weighted Average of Grade Points is also defined Grade Points are weighted by Credits and averaged over total credits in a Semester. This process is repeated for all Semesters and a CGPA defines the Final Academic Performance

15 What are the norms for the number of Credits per Semester and total number of Credits for UG/PG program?

These norms are usually defined by UGC or AICTE. Usually around 25 Credits per semester is the accepted norm.

16 What is a Semester Grade Point Average (SGPA)?

The performance of a student in a semester is indicated by a number called SGPA. The SGPA is the weighted average of the grade points obtained in all the courses registered by the student during the semester.

$$SGPA = \sum_{i=1}^n (C_i G_i) / \sum_{i=1}^n C_i$$

Where, C_i is the number of credits of the i^{th} course and G_i is the grade point scored by the student in the i^{th} course and i represent the number of courses in which a student registered in the concerned semester. SGPA is rounded to two decimal places.

17 What is a Cumulative Grade Point Average (CGPA)?

An up-to-date assessment of overall performance of a student from the time of his first registration is obtained by calculating a number called CGPA, which is weighted average of the grade points obtained in all the courses registered by the students since he entered the Institute.

$$CGPA = \sum_{j=1}^m (C_j S_j) / \sum_{j=1}^m C_j$$

Where, S_j is the SGPA of the j^{th} semester and C_j is the total number of credits upto the semester and m represent the number of semesters completed in which a student registered upto the semester. CGPA is rounded to two decimal places.

18 Is there any Software available for calculating Grade point averages and converting the same into Grades?

Yes, The institute has its own MIS software for calculation of SGPA, CGPA, etc.

19 Will the teacher be required to do the job of calculating SGPAs etc. and convert the same into Grades?

No. The teacher has to give marks obtained out of whatever maximum marks as it is. Rest is all done by the computer.

20 Will there be any Revaluation or Re-Examination System?

No. There will double valuation of answer scripts. There will be a makeup Examination after a reasonable preparation time after the End Semester Examination for specific cases mentioned in the Rules and Regulations. In addition to this, there shall be a 'summer term' (compressed term) followed by the End Semester Exam, to save the precious time of students.

21 How fast Syllabi can be and should be changed?

Autonomy allows us the freedom to change the syllabi as often as we need.

22 Will the Degree be awarded on the basis of only final year performance?

No. The CGPA will reflect the average performance of all the semester taken together.

23 What are Statutory Academic Bodies?

Governing Body, Academic Council, Examination Committee and Board of Studies are the different statutory bodies. The participation of external members in everybody is compulsory. The institute has nominated professors from IIT, NIT, University (the officers of the rank of Pro-vice Chancellor, Deans and Controller of Examinations) and also the reputed industrialist and industry experts on these bodies.

24 Who takes Decisions on Academic matters?

The Governing Body of institute is the top academic body and is responsible for all the academic decisions. Many decisions are also taken at the lower level like Boards of Studies. Decisions taken at the Board of Studies level are to be ratified at the Academic Council and Governing Body.

25 What is the role of Examination committee?

The Examinations Committee is responsible for the smooth conduct of internal, End Semester and make up Examinations. All matters involving the conduct of examinations spot valuations, tabulations preparation of Grade Sheet etc fall within the duties of the Examination Committee.

26 Is there any mechanism for Grievance Redressal?

The institute has grievance redressal committee, headed by Dean - Student affairs and Dean - IQAC.

27 How many attempts are permitted for obtaining a Degree?

All such matters are defined in Rules & Regulation.

28 Who declares the result?

The result declaration process is also defined. After tabulation work wherein the SGPA, CGPA and final Grades are ready, the entire result is reviewed by the Moderation Committee. Any unusual deviations or gross level discrepancies are deliberated and removed. The entire result is discussed in the Examinations and Result Committee for its approval. The result is then declared on the institute notice boards as well put on the web site and Students Corner. It is eventually sent to the University.

29 Who will keep the Student Academic Records, University or IARE?

It is the responsibility of the Examination Control Office of the institute to keep and preserve all the records.

30 What is our relationship with the JNT University?

We remain an affiliated college of the JNT University. The University has the right to nominate its members on the academic bodies of the college.

31 Shall we require University approval if we want to start any New Courses?

Yes, It is expected that approvals or such other matters from an autonomous college will receive priority.

32 Shall we get autonomy for PG and Doctoral Programs also?

Yes, presently our PG programs also enjoying autonomous status.

MALPRACTICE RULES

DISCIPLINARY ACTION FOR / IMPROPER CONDUCT IN EXAMINATIONS

S. No	Nature of Malpractices / Improper conduct	Punishment
	<i>If the candidate:</i>	
1. (a)	Possesses or keeps accessible in examination hall, any paper, note book, programmable calculator, cell phone, pager, palm computer or any other form of material concerned with or related to the course of the examination (theory or practical) in which he is appearing but has not made use of (material shall include any marks on the body of the candidate which can be used as an aid in the course of the examination)	Expulsion from the examination hall and cancellation of the performance in that course only.
(b)	Gives assistance or guidance or receives it from any other candidate orally or by any other body language methods or communicates through cell phones with any candidate or persons in or outside the exam hall in respect of any matter.	Expulsion from the examination hall and cancellation of the performance in that course only of all the candidates involved. In case of an outsider, he will be handed over to the police and a case is registered against him.
2.	Has copied in the examination hall from any paper, book, programmable calculators, palm computers or any other form of material relevant to the course of the examination (theory or practical) in which the candidate is appearing.	Expulsion from the examination hall and cancellation of the performance in that course and all other courses the candidate has already appeared including practical examinations and project work and shall not be permitted to appear for the remaining examinations of the courses of that Semester/year. The Hall Ticket of the candidate is to be cancelled and sent to the Controller of Examinations.
3.	Impersonates any other candidate in connection with the examination.	The candidate who has impersonated shall be expelled from examination hall. The candidate is also debarred and forfeits the seat. The performance of the original candidate, who has been impersonated, shall be cancelled in all the courses of the examination (including practicals and project work) already appeared and shall not be allowed to appear for examinations of the remaining courses of that semester/year. The candidate is also debarred for two consecutive semesters from class work and all semester end examinations. The continuation of the course by the candidate is course to the academic regulations in connection with forfeiture of seat. If the imposter is an outsider, he will be handed over to the police and a case is registered against him.
4.	Smuggles in the Answer book or additional sheet or takes out or arranges to send out the question paper during the examination or answer book or additional sheet, during or after the examination.	Expulsion from the examination hall and cancellation of performance in that course and all the other courses the candidate has already appeared including practical examinations and project work and shall not be permitted for the

		remaining examinations of the courses of that semester/year. The candidate is also debarred for two consecutive semesters from class work and all semester end examinations. The continuation of the course by the candidate is course to the academic regulations in connection with forfeiture of seat.
5.	Uses objectionable, abusive or offensive language in the answer paper or in letters to the examiners or writes to the examiner requesting him to award pass marks.	Cancellation of the performance in that course.
6.	Refuses to obey the orders of the Controller of Examinations /Additional Controller of Examinations/any officer on duty or misbehaves or creates disturbance of any kind in and around the examination hall or organizes a walk out or instigates others to walk out, or threatens the COE or any person on duty in or outside the examination hall of any injury to his person or to any of his relations whether by words, either spoken or written or by signs or by visible representation, assaults the COE or any person on duty in or outside the examination hall or any of his relations, or indulges in any other act of misconduct or mischief which result in damage to or destruction of property in the examination hall or any part of the Institute premises or engages in any other act which in the opinion of the officer on duty amounts to use of unfair means or misconduct or has the tendency to disrupt the orderly conduct of the examination.	In case of students of the college, they shall be expelled from examination halls and cancellation of their performance in that course and all other courses the candidate(s) has (have) already appeared and shall not be permitted to appear for the remaining examinations of the courses of that semester/year. The candidates also are debarred and forfeit their seats. In case of outsiders, they will be handed over to the police and a police case is registered against them.
7.	Leaves the exam hall taking away answer script or intentionally tears off the script or any part thereof inside or outside the examination hall.	Expulsion from the examination hall and cancellation of performance in that course and all the other courses the candidate has already appeared including practical examinations and project work and shall not be permitted for the remaining examinations of the courses of that semester/year. The candidate is also debarred for two consecutive semesters from class work and all semester end examinations. The continuation of the course by the candidate is course to the academic regulations in connection with forfeiture of seat.
8.	Possess any lethal weapon or firearm in the examination hall.	Expulsion from the examination hall and cancellation of the performance in that course and all other courses the candidate has already appeared including practical examinations and project work and shall not be permitted for the remaining examinations of the courses of that semester/year. The candidate is also debarred and forfeits the seat.
9.	If student of the college, who is not a candidate for the particular examination or any person not connected with the college indulges in any	Student of the colleges expulsion from the examination hall and cancellation of the performance in that course and all other

	malpractice or improper conduct mentioned in clause 6 to 8.	<p>courses the candidate has already appeared including practical examinations and project work and shall not be permitted for the remaining examinations of the courses of that semester/year. The candidate is also debarred and forfeits the seat.</p> <p>Person(s) who do not belong to the College will be handed over to police and, a police case will be registered against them.</p>
10.	Comes in a drunken condition to the examination hall.	Expulsion from the examination hall and cancellation of the performance in that course and all other courses the candidate has already appeared including practical examinations and project work and shall not be permitted for the remaining examinations of the courses of that semester/year.
11.	Copying detected on the basis of internal evidence, such as, during valuation or during special scrutiny.	Cancellation of the performance in that course and all other courses the candidate has appeared including practical examinations and project work of that semester/year examinations.
12.	If any malpractice is detected which is not covered in the above clauses 1 to 11 shall be reported to the University for further action to award suitable punishment.	

**FAILURE TO READ AND UNDERSTAND
THE REGULATIONS IS NOT AN EXCUSE**



INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal, Hyderabad - 500 043

COURSE CATALOGUE

REGULATIONS: BT-23

ELECTRICAL AND ELECTRONICS ENGINEERING

I SEMESTER

Course Code	Course Name	Subject Area	Category	Periods Per Week			Credits	Scheme of Examination Max. Marks		
				L	T	P		CIA	SEE	Total
INDUCTION PROGRAM				TWO WEEKS MANDATORY AUDIT COURSE						
THEORY										
AHSD01	Professional Communication	HSMC	Foundation	3	0	0	3	40	60	100
AHSD02	Matrices and Calculus	BSC	Foundation	3	1	0	4	40	60	100
AEED02	Electrical Circuits	ESC	Foundation	3	0	0	3	40	60	100
ACSD01	Object Oriented Programming	ESC	Foundation	3	0	0	3	40	60	100
PRACTICAL										
AHSD04	Professional Communication Laboratory	HSMC	Foundation	0	0	2	1	40	60	100
ACSD02	Object Oriented Programming with Java Laboratory	ESC	Foundation	0	1	2	2	40	60	100
AMED02	Manufacturing Practice	ESC	Foundation	1	0	2	2	40	60	100
AEED04	Electrical Circuits Laboratory	ESC	Foundation	0	0	2	1	40	60	100
EXPERIENTIAL ENGINEERING EDUCATION (ExEED)										
ACSD03	Essentials of Innovation	Skill	Skill	0	0	2	1	40	60	100
MANDATORY COURSE										
AHSD06	Environmental Science	MC	MC - I	Ref: Academic Regulations BT23						
TOTAL				13	02	10	20	360	540	900

II SEMESTER

Course Code	Course Name	Subject Area	Category	Periods Per Week			Credits	Scheme of Examination Max. Marks		
				L	T	P		CIA	SEE	Total
THEORY										
AHSD03	Engineering Chemistry	BSC	Foundation	3	0	0	3	40	60	100
AHSD07	Applied Physics	BSC	Foundation	3	0	0	3	40	60	100
AHSD08	Differential Equations and Vector Calculus	BSC	Foundation	3	1	0	4	40	60	100
ACSD05	Essentials of Problem Solving	ESC	Foundation	3	0	0	3	40	60	100
PRACTICAL										
AHSD05	Engineering Chemistry Laboratory	BSC	Foundation	0	0	2	1	40	60	100
AHSD09	Applied Physics Laboratory	BSC	Foundation	0	0	2	1	40	60	100
ACSD06	Programming for Problem Solving Laboratory	ESC	Foundation	0	1	2	2	40	60	100
AMED03	Engineering Graphics	ESC	Foundation	1	0	2	2	40	60	100
SKILL ENHANCEMENT PROJECT										
ACSD07	Mobile and Web Applications Development	Skill	Skill	0	0	2	1	40	60	100
MANDATORY COURSE										
AHSD10	Gender Sensitization	MC	MC - II	Ref: Academic Regulations BT-23						
FIELD PROJECT										
TOTAL				13	02	10	20	360	540	900

III SEMESTER

Course Code	Course Name	Subject Area	Category	Periods per week			Credits	Scheme of Examination Max. Marks		
				L	T	P		CIA	SEE	Total
THEORY										
AEED05	Analysis of Electrical Networks	PCC	Core	3	1	0	4	40	60	100
AEED06	DC Machines and Transformers	PCC	Core	3	0	0	3	40	60	100
AEED07	Electromagnetic Fields	PCC	Core	3	0	0	3	40	60	100
ACSD08	Data Structures	PCC	Core	3	0	0	3	40	60	100
AECD05	Analog Circuits	PCC	Core	3	0	0	3	40	60	100
PRACTICAL										
AEED08	Electrical Networks and Simulation Laboratory	PCC	Core	0	0	2	1	40	60	100
AEED09	DC Machines Laboratory	PCC	Core	0	0	2	1	40	60	100
ACSD11	Data Structures Laboratory	PCC	Core	0	0	2	1	40	60	100
EXPERIENTIAL ENGINEERING EDUCATION (ExEED)										
ACSD12	Prototype and Design Building	Skill	Skill	0	0	2	1	40	60	100
TOTAL				15	01	08	20	360	540	900

IV SEMESTER

Course Code	Course Name	Subject Area	Category	Periods per week			Credits	Scheme of Examination Max. Marks		
				L	T	P		CIA	SEE	Total
THEORY										
AEED10	AC Machines	PCC	Core	3	0	0	3	40	60	100
AEED11	Power Generation Systems and Economic Aspects	PCC	Core	3	0	0	3	40	60	100
AEED12	Control Systems	PCC	Core	3	0	0	3	40	60	100
AECD13	Digital Circuits	PCC	Core	3	0	0	3	40	60	100
AHSD13	Complex Analysis and Probability Distributions.	PCC	Core	3	0	0	3	40	60	100
PRACTICAL										
AEED13	AC Machines Laboratory	PCC	Core	0	0	2	1	40	60	100
AEED14	Control Systems Laboratory	PCC	Core	0	0	2	1	40	60	100
AECD17	Analog and Digital Circuits Laboratory	PCC	Core	0	0	2	1	40	60	100
SKILL ENHANCEMENT PROJECT										
ACSD18	DevOps Engineering	Skill	Skill	1	0	2	2	40	60	100
VALUE ADDED COURSE										
INTERNSHIP										
TOTAL				16	00	08	20	360	540	900

#The course would consist of talks by working professionals from industry, government, academia & research organizations.

V SEMESTER

Course Code	Course Name	Subject Area	Category	Periods per week			Credits	Scheme of Examination Max. Marks		
				L	T	P		CIA	SEE	Total
THEORY										
AEED16	Electrical Measurements and Instrumentation	PCC	Core	3	0	0	3	40	60	100
AEED17	Power Electronics	PCC	Core	3	0	0	3	40	60	100
AECD19	Microprocessors and Microcontrollers	PCC	Core	3	0	0	3	40	60	100
AEED18	Performance of Tranmision Lines and Cables	PCC	Core	3	0	0	3	40	60	100
	Program Elective – I	PEC	Elective	3	0	0	3	40	60	100
PRACTICAL										
AEED24	Electrical Measurements and Instrumentation Laboratory	PCC	Core	0	0	2	1	40	60	100
AECD28	Microprocessors and Microcontrollers Laboratory	PCC	Core	0	0	2	1	40	60	100
SKILL ENHANCEMENT PROJECT										
ACSD29	Engineering Design Project	Skill	Skill	0	0	2	1	40	60	100
ACSD30	Java Full Stack Development	Skill	Skill	1	0	2	2	40	60	100
VALUE ADDED COURSE										
TOTAL				16	0	08	20	360	540	900

#The course would consist of talks by working professionals from industry, government, academia & research organizations.

VI SEMESTER

Course Code	Course Name	Subject Area	Category	Periods per week			Credits	Scheme of Examination Max. Marks		
				L	T	P		CIA	SEE	Total
THEORY										
AEED25	Power System Analysis	PCC	Core	3	0	0	3	40	60	100
AEED26	Electric Drives and Static Control	PCC	Core	3	0	0	3	40	60	100
	Program Elective – II	PEC	Elective	3	0	0	3	40	60	100
	Program Elective – III	PEC	Elective	3	0	0	3	40	60	100
	Open Elective – I	OEC	Elective	3	0	0	3	40	60	100
PRACTICAL										
AEED39	Power Electronics and Drives Laboratory	PCC	Core	0	0	2	1	40	60	100
AEED40	PLC and Industrial Automation Laboratory	PCC	Core	0	0	2	1	40	60	100
SKILL ENHANCEMENT PROJECT										
ACSD44	Data Scientist / AI Specialist	Skill	Skill	1	0	2	2	40	60	100
ACSD45	Development Project	Skill	Skill	0	0	2	1	40	60	100
VALUE ADDED COURSE										
INTERNSHIP										
TOTAL				16	00	08	20	360	540	900

#The course would consist of talks by working professionals from industry, government, academia & research organizations.

VII SEMESTER

Course Code	Course Name	Subject Area	Category	Periods per week			Credits	Scheme of Examination Max. Marks		
				L	T	P		CIA	SEE	Total
THEORY										
AEED41	Operation and Control of Electrical Power Systems	PCC	Core	3	0	0	3	40	60	100
AEED42	Switch Gear and Protection of Electrical Apparatus	PCC	Core	3	0	0	3	40	60	100
	Program Elective - IV	PEC	Elective	3	0	0	3	40	60	100
	Program Elective - V	PEC	Elective	3	0	0	3	40	60	100
	Open Elective - II	OEC	Elective	3	0	0	3	40	60	100
PRACTICAL										
AEED55	Power System Simulation Laboratory	PCC	Core	0	0	2	1	40	60	100
AEED56	Power System Protection Laboratory	PCC	Core	0	0	2	1	40	60	100
PROJECT WORK										
AEED57	Project Work (Phase - I)	PROJ	Project	0	0	6	3	40	60	100
MANDATORY COURSE										
AHSD14	Essence of Indian Traditional Knowledge	MC	MC-III	Ref: Academic Regulations BT23						
TOTAL				15	00	10	20	320	480	800

VIII SEMESTER

Course Code	Course Name	Subject Area	Category	Periods per week			Credits	Scheme of Examination Max. Marks		
				L	T	P		CIA	SEE	Total
THEORY										
AHSD15	Managerial Economics and Financial Analysis	BSC	Foundation	3	0	0	3	40	60	100
	Program Elective - VI	PEC	Elective	3	0	0	3	40	60	100
	Open Elective-III	OEC	Elective	3	0	0	3	40	60	100
PROJECT WORK										
AEED65	Project Work (Phase - II)	PCC	Project	0	0	22	11	40	60	100
TOTAL				09	00	22	20	160	240	400

ELECTIVE COURSES

PROGRAMME ELECTIVE COURSES (PE)

Course Code	Course Name	Prerequisites	Preferred semester	Credits
AEED19	Computer Aided Electrical Machines Design	DCMT and AC Machines	V	3
AEED20	Special Electrical Machines	DCMT and AC Machines	V	3
AEED21	High Voltage Engineering	LAC	V	3
AEED22	Electrical Energy Conservation and Auditing	Power System	V	3
AEED23	IOT Architecture and Protocols	-	V	3
AEED27	Principles of Signals and Systems	Complex Analysis and Transform Techniques	VI	3
AEED28	HVDC Transmission	EMI	VI	3
AEED29	Power System Stability	Power System	VI	3
AEED30	Digital Protection of Power System	Power System	VI	3
AEED31	Swarm Intelligence Techniques in Electrical Engineering	Power System	VI	3
AEED32	Power System Dynamics and Control	Power System	VI	3
AEED33	Engineering Optimization	Power System	VI	3
AEED34	EHVAC Transmission	Power System	VI	3
AEED35	Cyber Security in Power Systems	Power System	VI	3
AEED36	Sustainable Energy Systems	Energy Systems and Sustainability	VI	3
AEED43	Grid Instrumentation and Communication	Power System	VII	3
AEED44	Power Quality and FACTS	Power System	VII	3
AEED45	Data Science and Machine Learning in Electrical Engineering	Power System	VII	3
AEED46	Battery Management Systems	Energy Systems and Sustainability	VII	3
AEED47	Utilization of Electric Power	Electrical Machine	VII	3
AEED48	Industrial Electrical Systems	-	VII	3
AEED49	Industrial IOT and Automation	Control System	VII	3
AEED50	Artificial Intelligence and Robotics	Electrical Machine	VII	3
AEED51	Electrical Traction	Power Electronics	VII	3
AEED52	Advanced Instrumentation	EMI	VIII	3
AEED58	Modelling, Dynamics and control of EVs	Electrical Machine	VIII	3
AEED59	Small Signal Analysis of Power Converters	Power Electronics	VIII	3
AEED60	Advanced Power Systems	Instrumentation	VIII	3
AEED61	Robotics and Mechatronics	Control Systems	VIII	3
AEED62	Process Control and Instrumentation	Control Systems	VIII	3

OPEN ELECTIVE COURSES (OEC)

The courses listed below are offered by the department of Electrical and Electronics Engineering for **students of other departments.**

Course Code	Course Name	Credits
AEED37	Electrical Estimation and Installation	3
AEED38	Utilization of Electrical Engineering	3
AEED53	Electric Power generation and Distribution	3
AEED54	Renewable Energy Sources	3
AEED63	Electric Vehicle	3
AEED64	Battery Technology	3

COURSE CONTENT

PROFESSIONAL COMMUNICATION								
I Semester: AE / ME / CE / CSE (AI &ML) / IT / ECE / EEE								
II Semester: CSE CSE (DS) CSE (CS)								
Course Code	Category	Hours / Week			Credits	Maximum Marks		
AHSD01	Foundation	L	T	P	C	CIA	SEE	Total
		3	-	-	3	40	60	100
Contact Classes: 64	Tutorial Classes: Nil	Practical Classes: Nil			Total Classes: 48			
Prerequisite:								

I. COURSE OVERVIEW:

The principle aim of the course is that the students will have awareness about the importance of English language in the contemporary times and also it emphasizes the students to learn this language as a skill (listening skill, speaking skill, reading skill and writing skill). Moreover, the course benefits the students how to solve their day-to-day problems in speaking English language. Besides, it assists the students to reduce the mother tongue influence and acquire the knowledge of neutral accent. The course provides theoretical and practical knowledge of English language and it enables students to participate in debates about informative, persuasive, didactic, and commercial purposes.

II. COURSE OBJECTIVES:

The students will try to learn:

- I Standard pronunciation, appropriate word stress, and necessary intonation patterns for effective communication towards achieving academic and professional targets.
- II Appropriate grammatical structures and also using the nuances of punctuation tools for practical purposes.
- III Critical aspect of speaking and reading for interpreting in-depth meaning between the sentences.
- IV Conceptual awareness on writing in terms of unity, content, coherence, and linguistic accuracy.

III. COURSE OUTCOMES:

After successful completion of the course, students should be able to:

- CO 1 Demonstrate the prime necessities of listening skills and communication skills for academic and non-academic purposes.
- CO 2 Explain effectively in spoken English on issues and ideas with a reasonable degree of fluency and accuracy in different social settings and different kinds of social encounters.
- CO 3 Strengthen acceptable language for developing life skills to overcome the challenges at professional platform.
- CO 4 Interpret the grammatical and lexical forms of English and use of these forms in specific communicative contexts.
- CO 5 Articulate main ideas, both stated and inferred, and important details in academic, journalistic, and literary prose at advanced level.
- CO 6 Extend writing skills for fulfilling academic and work-place requirements of various written communicative functions.

IV. COURSE CONTENT:

MODULE – I: GENERAL INTRODUCTION AND LISTENING SKILLS (13)

Introduction to communication skills, communication process, elements of communication, soft skills and hard skills, importance of soft skills for engineers, listening skills, significance of listening skills, stages of listening, barriers and effectiveness of listening, listening comprehension.

MODULE – II: SPEAKING SKILL (13)

Significance of speaking skills, essentials of speaking skills, verbal and non-verbal communication, generating talks based on visual prompts, public speaking, exposure to structured talks, delivering speech effectively, oral presentation using power point slides.

MODULE – III: VOCABULARY AND GRAMMAR (13)

The concept of word formation, idioms and phrases, one-word substitutes, sentence structure (simple, compound and complex), usage of punctuation marks, advanced level prepositions.

Tenses, subject verb agreement, degrees of comparison, direct and indirect speech, questions tags.

MODULE – IV: READING SKILL (12)

Significance of reading skills, techniques of reading, skimming-reading for the gist of a text, scanning-reading for specific information, intensive, extensive reading, reading comprehension, metaphor and figurative language.

MODULE – V: WRITING SKILL (13)

Significance of writing skills, effectiveness of writing, the role of a topic sentence and supporting sentences in a paragraph, organizing principles of paragraphs in a document, writing introduction and conclusion, techniques for writing precis, various formats for letter writing (block format, full block format, and semi block format), e-mail writing, report writing.

V. TEXT BOOKS:

1. Anjana Tiwari, *Communication Skills in English*, Khanna Publishing House: New Delhi, 2022.

VI. REFERENCE BOOKS:

1. Norman Whitby, *Business Benchmark: Pre-Intermediate to Intermediate – BEC Preliminary*, Cambridge University Press, 2nd Edition, 2008.
2. Devaki Reddy, Shreesh Chaudhary, *Technical English*, Macmillan, 1st Edition, 2009.
3. Rutherford, Andrea J, *Basic Communication Skills for Technology*, Pearson Education, 2nd Edition, 2010.
4. Raymond Murphy, *Essential English Grammar with Answers*, Cambridge University Press, 2nd Edition, 2010.

VII. ELECTRONIC RESOURCES:

1. https://akanksha.iare.ac.in/index?route=course/details&course_id=954
2. https://akanksha.iare.ac.in/index?route=course/details&course_id=10
3. https://akanksha.iare.ac.in/index?route=course/details&course_id=352
4. <https://akanksha.iareac.in/index?route=publicprofile&id=5075>

VIII. MATERIALS ONLINE

1. Course template
2. Tech-talk topics
3. Assignments
4. Definition and terminology
5. Tutorial question bank
6. Model question paper – I
7. Model question paper – II
8. Lecture notes
9. Early lecture readiness videos (ELRV)
10. Power point presentations

COURSE CONTENT

MATRICES AND CALCULUS								
I Semester: AE / ME / CE / ECE / EEE / CSE / CSE (AI & ML) / CSE (DS) / CSE (CS) / IT								
Course Code	Category	Hours/Week			Credits	Maximum Marks		
AHSD02	Foundation	L	T	P	C	CIA	SEE	Total
		3	1	-	4	40	60	100
Contact Classes: 48	Tutorial Classes: 16	Practical Classes: Nil			Total Classes: 64			
Prerequisite: Basic Principles of Algebra and Calculus								

I. COURSE OVERVIEW:

This course Matrices and Calculus is a foundation course of mathematics for all engineering branches. The concepts of Matrices, Eigen Values, Eigen Vectors, Functions of Single and Several Variables, Fourier Series and Multiple Integrals. This course is applicable for simulations, colour imaging process, finding optimal solutions in all fields of industries.

II. COURSE OBJECTIVES:

The students will try to learn:

- I The concept of the rank of a matrix, solve the system of linear equations, eigen values, eigen vectors.
- II The geometrical approach to the mean value theorems and their application to the mathematical problems.
- III The Fourier series expansion in standard intervals as well as arbitrary intervals.
- IV The evaluation of multiple integrals and their applications.

III. COURSE OUTCOMES:

At the end of the course students should be able to:

- CO 1 Determine the rank and solutions of linear equations with elementary operations.
- CO 2 Utilize the Eigen values, Eigen vectors for developing spectral matrices.
- CO 3 Make use of Cayley-Hamilton theorem for finding powers of the matrix.
- CO 4 Interpret the maxima and minima of given functions by finding the partial derivatives.
- CO 5 Apply the Fourier series expansion of periodic functions for analyzing the wave forms.
- CO 6 Determine the area of solid bounded regions by using the integral calculus.

IV. COURSE CONTENT:

MODULE - I: MATRICES (09)

Rank of a matrix by echelon form and normal form, inverse of non-singular matrices by Gauss-Jordan method, system of linear equations, solving system of homogeneous and non-homogeneous equations.

MODULE - II: EIGEN VALUES AND EIGEN VECTORS (10)

Eigen values, eigen vectors and their properties (without proof), Cayley-Hamilton theorem (without proof), verification, finding inverse and power of a matrix by Cayley-Hamilton theorem, diagonalization of a matrix.

MODULE - III: FUNCTIONS OF SINGLE and SEVERAL VARIABLES (10)

Mean value theorems Rolle's theorem, Lagrange's theorem, Cauchy's theorem-without proof.

Functions of several variables: Partial differentiation, Jacobian, functional dependence, maxima and minima of functions of two variables and three variables, method of Lagrange multipliers.

MODULE – IV: FOURIER SERIES (09)

Fourier expansion of periodic function in a given interval of length 2π , Fourier series of even and odd functions, Fourier series in an arbitrary interval, half- range Fourier sine and cosine expansions.

MODULE – V: MULTIPLE INTEGRALS (10)

Evaluation of double integrals (cartesian and polar coordinates), change of order of integration (only cartesian coordinates), evaluation of triple integrals (only cartesian coordinates).

V. TEXT BOOKS:

1. B. S. Grewal, *Higher Engineering Mathematics*, 44/e, Khanna Publishers, 2017.
2. Erwin Kreyszig, *Advanced Engineering Mathematics*, 10/e, John Wiley & Sons, 2011.

VI. REFERENCE BOOKS:

1. R. K. Jain and S. R. K. Iyengar, *Advanced Engineering Mathematics*, 3/ed, Narosa Publications, 5th Edition, 2016.
2. George B. Thomas, Maurice D. Weir and Joel Hass, Thomas, *Calculus*, 13/e, Pearson Publishers, 2013.
3. N.P. Bali and Manish Goyal, *A text book of Engineering Mathematics*, Laxmi Publications, Reprint, 2008.
4. Dean G. Duffy, *Advanced Engineering Mathematics with MATLAB*, CRC Press.
5. Peter O’Neil, *Advanced Engineering Mathematics*, Cengage Learning.
6. B.V. Ramana, *Higher Engineering Mathematics*, McGraw Hill Education.

VII. ELECTRONIC RESOURCES:

1. Engineering Mathematics - I, By Prof. Jitendra Kumar | IIT Kharagpur
https://onlinecourses.nptel.ac.in/noc23_ma88/preview
2. Advanced Calculus for Engineers, By Prof. Jitendra Kumar, Prof. Somesh Kumar | IIT Kharagpur
https://onlinecourses.nptel.ac.in/noc23_ma86/preview
3. http://www.efunda.com/math/math_home/math.cfm
4. <http://www.ocw.mit.edu/resources/#Mathematics>
5. <http://www.sosmath.com>
6. <http://www.mathworld.wolfram.com>

VIII. MATERIAL ONLINE:

1. Course template
2. Tech-talk topics
3. Assignments
4. Definition and terminology
5. Tutorial question bank
6. Model question paper – I
7. Model question paper – II
8. Lecture notes
9. Early lecture readiness videos (ELRV)
10. Power point presentations



INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal - 500 043, Hyderabad, Telangana

COURSE CONTENT

ELECTRICAL CIRCUITS								
I Semester: EEE / ECE								
Course Code	Category	Hours / Week			Credits	Maximum Marks		
AEED02	Foundation	L	T	P	C	CIA	SEE	Total
		3	-	-	3	40	60	100
Contact Classes: 48	Tutorial Classes: Nil	Practical Classes: Nil			Total Classes: 48			
Prerequisite: There are no prerequisites to take this course.								

I. COURSE OVERVIEW:

The course introduces the basic concepts of circuit analysis which is the foundation for all subjects of the electrical and electronics engineering. It includes the basic fundamental laws of electricity and magnetism with an emphasis on resistors, inductors and capacitors (RLC) circuits applied to alternating current (AC) or direct current (DC) of electrical networks. Further This course provides network theorems with different excitations, two-port network and network topology to solve for real-time applications.

II. COURSE OBJECTIVES:

The students will try to learn

- The network reduction techniques such as source transformation, mesh analysis, nodal analysis and network theorems to solve different networks.
- The basic concept of AC circuits for optimization of household and industrial circuitry.
- The various configurations of electromagnetic induction used in magnetic circuits helps in the winding of electrical machines.
- The characteristics of two-port networks and network topologies suitable in power system.

III. COURSE OUTCOMES:

At the end of the course students should be able to:

- | | |
|------|--|
| CO 1 | Define the various terminology used to study the characteristics of DC and AC electrical networks. |
| CO 2 | Demonstrate the different laws and indirect quantities associated with electrical circuit for branch currents and nodal voltages |
| CO 3 | Summarize the procedure for several theorems to reduce complex network into simple equivalent network with DC and AC excitation. |
| CO 4 | Describe the electromagnetic induction, magnetic flux, self and mutual inductance in the single coil and coupled coils magnetic circuits to know total magneto motive force and total ampere turns values. |
| CO 5 | Develop the two port parameters and network topology for graphical and digital representation of complex circuits to be measured easily, without solving for all the internal voltages and currents in the different networks. |
| CO 6 | Illustrate the importance of dual network for comparing both mesh and nodal networks. |

IV. COURSE CONTENT:

MODULE – I: INTRODUCTION TO ELECTRICAL CIRCUITS (09)

Circuit concept: Basic definitions, Ohm's law at constant temperature, classifications of elements, independent and dependent sources, voltage and current relationships for passive elements.

Single phase AC circuits: Representation of alternating quantities, properties of different periodic wave forms, phase and phase difference, concept of impedance and admittance, power in AC circuits.

MODULE – II: ANALYSIS OF ELECTRICAL CIRCUITS (09)

Circuit analysis: Source transformation, Kirchhoff's laws, total resistance, inductance and capacitance of circuits, Star - delta transformation technique, mesh analysis and nodal analysis, inspection method, super mesh, super node analysis.

MODULE – III: NETWORK THEOREMS (DC AND AC) (10)

Network Theorems: Tellegen's, superposition, reciprocity, Thevenin's, Norton's, maximum power transfer, Milliman's and compensation theorems for DC excitations, numerical problems.

Network Theorems: Tellegen's, superposition, reciprocity, Thevenin's, Norton's, maximum power transfer, Milliman's and compensation theorems for AC excitations, numerical problems.

MODULE – IV: MAGNETIC CIRCUITS (10)

Magnetic circuits: Faraday's laws of electromagnetic induction, concept of self and mutual inductance, dot convention, coefficient of coupling, composite magnetic circuit, analysis of series and parallel magnetic circuits.

MODULE – V: TWO PORT NETWORK AND GRAPH THEORY (10)

Two Port Network: Two port parameters, interrelations, Two port Interconnections.

Network topology: Definitions, incidence matrix, basic tie set and basic cut set matrices for planar networks, duality and dual networks.

V. TEXT BOOKS:

1. A Sudhakar, Shyam mohan S Palli, "Circuits and Networks", Tata McGraw-Hill, 4th Edition, 2010.
2. M E Van Valkenberg, "Network Analysis", PHI, 3rd Edition, 2014.

VI. REFERENCE BOOKS:

1. John Bird, *Electrical Circuit Theory and Technology*, Newnes, 2nd Edition, 2003.
2. C L Wadhwa, *Electrical Circuit Analysis including Passive Network Synthesis*, New Age International, 2nd Edition, 2009.
3. David A Bell, *Electric Circuits*, Oxford University Press, 7th Edition, 2009.
4. E Hughes, *Electrical and Electronics Technology*, Pearson Education, 2010.
5. A Chakrabarthy, *Electric Circuits*, Dhanipat Rai & Sons, 6th Edition, 2010.
6. V D Toro, *Electrical Engineering Fundamentals*, Prentice Hall India, 1989.

VII. WEB REFERENCES:

1. <https://www.igniteengineers.com>
2. <https://www.ocw.nthu.edu.tw>
3. <https://www.uotechnology.edu.iq>
4. <https://www.iare.ac.in>

VIII. MATERIALS ONLINE

1. Course template
2. Tutorial question bank
3. Tech talk topics
4. Assignments
5. Definitions and terminology
6. Open ended experiments
7. Model question paper-I
8. Model question paper-II
9. Lecture notes
10. Power point presentations
11. ELRV videos

COURSE CONTENT

OBJECT ORIENTED PROGRAMMING								
I Semester: AE / ME / CE / ECE / EEE / CSE / CSE (AI & ML) / CSE (DS) / CSE (CS) / IT								
Course Code	Category	Hours / Week			Credits	Maximum Marks		
ACSD01	Foundation	L	T	P	C	CIA	SEE	Total
		3	0	0	3	40	60	100
Contact Classes: 48	Tutorial Classes: Nil	Practical Classes: Nil			Total Classes: 48			
Prerequisites: There are no prerequisites to take this course.								

I. COURSE OVERVIEW:

The course provides a solid foundation in object-oriented programming concepts in using them. It includes concepts object-oriented concepts such as information hiding, encapsulation, and polymorphism. It contrasts the use of inheritance and composition as techniques for software reuse. It provides an understanding of object-oriented design using graphical design notations such as Unified Modeling Language (UML) as well as object design patterns.

II. COURSES OBJECTIVES:

The students will try to learn

- I. The fundamental concepts and principles of object-oriented programming in high-level programming languages.
- II. The advanced concepts for developing well-structured and efficient programs that involve complex data structures, numerical computations, or domain-specific operations.
- III. The design and implementation of features such as inheritance, polymorphism, and encapsulation for tackling complex problems and creating well-organized, modular, and maintainable code.
- IV. The usage of input/output interfaces to transmit and receive data to solve real-time computing problems.

III. COURSE OUTCOMES:

At the end of the course, students should be able to:

- CO 1 Interpret the features of object-oriented programming languages, comparison, and evaluation of programming languages.
- CO 2 Model the real-world scenario using class diagrams and exhibit communication between objects.
- CO 3 Estimate the need for special functions for data initialization.
- CO 4 Outline the features of object-oriented programming for binding the attributes and behavior of a real-world entity.
- CO 5 Use the concepts of streams and files that enable data management to enhance programming skills.
- CO 6 Develop contemporary solutions to software design problems using object-oriented principles.

IV. COURSE CONTENT:

MODULE - I: Object-oriented concepts (09)

Objects and legacy systems, procedural versus Object-oriented programming, top-down and bottom-up approaches and their differences, benefits of OOP, applications of OOP, features of OOP.

Abstraction: Layers of abstraction, forms of abstraction, abstraction mechanisms.

MODULE - II: Classes and objects (09)

Classes and objects: Object data, object behaviors, creating objects, attributes, methods, messages, creating class diagrams.

Access specifiers and initialization of class members: Accessing members and methods, access specifiers - public, private, protected, memory allocation. Static members, static methods.

MODULE - III: Special member functions and overloading (09)

Constructors and destructors: Need for constructors and destructors, copy constructors, dynamic constructors, parameterized constructors, destructors, constructors and destructors with static members.

Overloading: Function overloading, constructor overloading, and operator overloading - rules for overloading

operators, overloading unary and binary operators, friend functions.

MODULE – IV: Inheritance and polymorphism (09)

Inheritance: types of inheritance, base class, derived class, usage of final, ambiguity in multiple and multipath inheritance, virtual base class, overriding member functions, order of execution of constructors and destructors.

Polymorphism and virtual functions: Virtual functions, pure virtual functions, abstract classes, introduction to polymorphism, static polymorphism, dynamic polymorphism.

MODULE –V: Console I/O and working with files (09)

Console I/O: Concept of streams, hierarchy of console stream classes, unformatted I/O operations, managing output with manipulators.

Working with files: Opening, reading, writing, appending, processing, and closing different types of files, command line arguments.

V. TEXTBOOKS:

1. Matt Weisfeld, *The Object-Oriented Thought Process*, Addison Wesley Object Technology Series, 4th Edition, 2013.

VI. REFERENCE BOOKS:

1. Timothy Budd, *Introduction to object-oriented programming*, Addison Wesley Object Technology Series, 3rd Edition, 2002.
2. Gaston C. Hillar, *Learning Object-Oriented Programming*, Packt Publishing, 2015.
3. Kingsley Sage, *Concise Guide to Object-Oriented Programming*, Springer International Publishing, 1st Edition, 2019.
4. Rudolf Pecinovsky, *OOP - Learn Object Oriented Thinking and Programming*, Tomas Bruckner, 2013.
5. Grady Booch, *Object-oriented analysis and design with applications*, Addison Wesley Object Technology Series, 3rd Edition, 2007.

VII. ELECTRONIC RESOURCES:

1. <https://docs.oracle.com/javase/tutorial/java/concepts/>
2. <https://www.w3schools.com/cpp/>
3. <https://www.edx.org/learn/object-oriented-programming/>
4. <https://www.geeksforgeeks.org/introduction-of-object-oriented-programming/>

VIII. MATERIALS ONLINE

1. Course template
2. Tutorial question bank
3. Tech talk topics
4. Open-ended experiments
5. Definitions and terminology
6. Assignments
7. Model question paper – I
8. Model question paper – II
9. Lecture notes
10. PowerPoint presentation
11. E-Learning Readiness Videos (ELRV)



INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal - 500 043, Hyderabad, Telangana

COURSE CONTENT

PROFESSIONAL COMMUNICATION LABORATORY								
I Semester: CSE (AI&ML) IT AE ECE EEE ME CE								
II Semester: CSE CSE(DS) CSE(CS)								
Course Code	Category	Hours / Week			Credits	Maximum Marks		
AHSD04	Foundation	L	T	P	C	CIA	SEE	Total
		-	-	2	1	40	60	100
Contact Classes: Nil	Tutorial Classes: Nil	Practical Classes: 45			Total Classes: 45			
Prerequisite: Nil								

I. COURSE OVERVIEW:

This laboratory course is designed to introduce students to create a wide exposure on language learning techniques of the basic elements of listening skills, speaking skills, reading skills and writing skills. In this laboratory, students are trained in communicative English language skills, phonetics, word accent, word stress, rhythm, intonation, oral presentations and extempore speeches. Students are also taught in terms of seminars, group-discussions, presenting techniques of writing, participating in role plays, telephonic etiquettes, asking and giving directions, information transfer, debates, description of persons, places and objects etc. The laboratory encourages students to work in a group, engage in peer-reviews and inculcate team spirit through various exercises on grammar, vocabulary, and pronunciation games etc. Students will make use of all these language skills in academic, professional and real time situations.

II. COURSES OBJECTIVES

The students will try to learn:

- I. English speech sounds, word accent, intonation and stress patterns for effective pronunciation.
- II. Critical aspect of speaking and reading for interpreting in-depth meaning between the sentences.
- III. Language techniques for social interactions such as public speaking, group discussions and interviews.
- IV. Computer-assisted multi-media instructions and independent language learning.

III. COURSE OUTCOMES:

At the end of the course, students will be able to:

- CO 1 Articulate acceptable accent in order to execute formal and informal communication.
- CO 2 Differentiate stress shifts, syllabification and make use of past tense and plural markers effectively in connected speech; besides participate in role plays with confidence.
- CO 3 Use weak forms and strong forms in spoken language and maintain intonation patterns as a native speaker to avoid mother tongue influence; moreover, practice various etiquettes at professional platform.
- CO 4 Demonstrate errors in pronunciation and the decorum of oral presentations; for that reason, take part joining in group discussions and debates with much critical observations.
- CO 5 Strengthen writing effective messages, notices, summaries and also able to write reviews very critically of art and academical videos.
- CO 6 Argue scholarly, giving the counters to open ended experiments, and also writing slogans for the products talentedly.

Dos

1. Turn up in a neat and formal dress code regularly and maintain punctuality.
2. Bring observation books and worksheets for every laboratory session without fail.
3. Keep lab record book up to date.
4. Students must adhere to the acceptable use of ICT resources policy.
5. CD ROM 's, USB and other multimedia equipment are for college use only.
6. Replace headsets onto the monitor and rearrange your chairs back into their position as you leave laboratory.
7. Get your lab worksheets evaluated and upload online within the stipulated time for online evaluation by the faculty concerned.
8. Conduct yourself at the best to be a good learner.

Don'ts

1. Do not use the on/off switch to reboot the system.
2. Do not breach copyright regulations.
3. Do not install or download any software or modify or delete any system files on any laboratory computer.
4. Do not read or modify other users' files.
5. Do not damage, remove, or disconnect any labels, parts, cables or equipment.
6. Do not make undue noise in the laboratories. Be considerate of the other lab users- this is study area.
7. No food and the beverages are allowed into computer laboratories.

IV. COURSE CONTENT

Exercises for professional communication laboratory

Note: Students are encouraged to bring their own laptops for laboratory practice session

Getting started exercises

1. Introduction to pronunciation

Experiments

CALL LAB: Introduction to pronunciation

ICS LAB: Introducing self and introducing others and feedback

Activities

- a) Pronunciation practice among groups.
- b) Follow formal rules of introducing self and others and giving the constructive feedback.
- c) Pronunciation practice from the K-Van software

2. Introduction to phonetics

Experiments

CALL LAB: Introduction to phonetics, listening to English sounds, Vowel and Consonant sounds.

ICS LAB: Describing a person or place or a thing using relevant adjectives – feedback

Activities

- a) Introduction to phonetics, listening to English sounds, Vowel and Consonant sounds
- b) Describing a person or place or a thing using relevant adjectives and giving valid feedback.
- c) Listening English phonemes from K-Van software

3. Syllabification of different lexemes

Experiments

CALL LAB: Structure of syllables.

ICS LAB: JAM Sessions using public address system

Activities

- a) Participating in just a minute session
- b) Practicing consonant clusters
- c) Practicing different methods of dividing the syllables

4. Word accent and stress shift patterns

Experiments:

CALL LAB: Word accent and stress shifts.

ICS LAB: Asking for directions and giving directions

Activities:

- a) Usage of appropriate lexical tools at the time of giving directions
- b) Pronounce weak and strong forms using strategies in spoken language

5. Usage of markers in pronunciation

Experiments:

CALL LAB: Past tense and plural markers.

ICS LAB: Role plays on fixed expressions in various situations

Activities:

- a) Addition of suffixes to verbs

- b) Multiple exercises on nouns based its number
- c) Execution of body language in different contexts

6. Use of form in connected speech

CALL LAB: Weak forms and strong forms

ICS LAB: Extempore-Picture

Activities:

- a) Practice on recognition of objects
- b) Preparation on connected speech based on tone boundaries
- c) Execution while describing pictures

7. Intonation patterns

Experiments

CALL LAB: Intonation

ICS LAB: Interpretation of Proverbs and Idioms

Activities

- a) Voce modulation in speech as per context
- b) Application of proverbs in real life contexts
- c) Exercises on idiomatic expressions

8. Neutralization of Mother Tongue influence (MTI)

Experiments

CALL LAB: Neutralization of Mother Tongue Influence (MTI).

ICS LAB: Etiquette

Activities

- a) Mirror practicing to get rid of mother tongue influence
- b) Listening to English speeches from native speakers
- c) Etiquettes for social and professional behavior

9. Oral presentations

Experiments

CALL LAB: Common errors in pronunciation practice through tongue twisters.

ICS LAB: Oral Presentations

Activities

- a) Practicing tongue twisters to accelerate language fluency
- b) Practice on how to grab attention of audience
- c) Formulating strategies for structured presentation

10. Minimal pairs and debates

Experiments

CALL LAB: Minimal pairs.

ICS LAB: Debates

Activities

- a) Practicing minimal pair dominoes
- b) Working on minimal pair-cards and counters
- c) Executing interpersonal skills with argumentative issues

11. Listening comprehension skills

Experiments

CALL LAB: Listening comprehension.

ICS LAB: Group discussion

Activities

- a) Listening to dialogues of native speakers

- b) Listening and writing short stories
- c) Exercising group discussion on various social issues

12. Enriching writing skills

Experiments

CALL LAB: Demonstration on how to write leaflets, messages and notices.

ICS LAB: Techniques and methods to write summaries and reviews of videos

Activities

- a) Practice on writing suitable messages and notices
- b) Summary writing techniques
- d) Practice on writing reviews for sociopolitical videos

13. Activities on pronunciation skills

Experiments

CALL LAB: Pronunciation practice.

ICS LAB: Information transfer

Activities

- a) Exercise on commonly mispronounced words
- b) Preparation on similar words pronunciation
- c) Practice on transferring information through charts and diagrams

14. Writing reviews after watching statements

Experiments

CALL LAB: Open ended experiments-phonetics practice.

ICS LAB: Providing reviews and remarks

Activities

- a) Writing three term labels for vowels
- b) Writing three term labels for consonants
- c) Practice on giving reviews and remarks

15. Experiments on text to speech and writing slogans

Experiments

CALL LAB: Open-ended experiments-text to speech.

ICS LAB: Writing slogan related to the image

Activities

- a) Practice through streaming video lessons
- b) Training on creating dialogues and stories
- c) Practice on designing and writing slogans

V. TEXT BOOKS:

1. Professional Communication laboratory manual

VI. REFERENCE BOOK

1. Meenakshi Raman, Sangeetha Sharma, *Technical Communication Principles and Practices*, Oxford University Press, New Delhi, 3rd Edition, 2015.
2. Rhirdion, Daniel, *Technical Communication*, Cengage Learning, New Delhi, 1st Edition, 2009.

VII. ELECTRONIC RESOURCES

1. Cambridge online pronunciation dictionary <https://dictionary.cambridge.org/>
2. Fluentu website <https://www.fluentu.com/>
3. Repeat after us <https://brycs.org/clearinghouse/3018/>
4. Language lab <https://brycs.org/clearinghouse/3018/>
5. Oxford online videos

VIII. MATERIALS ONLINE

1. Course template
2. Lab manual

COURSE CONTENT

OBJECT ORIENTED PROGRAMMING WITH JAVA LABORATORY								
I Semester: AE / ME / CE / ECE / EEE / CSE / CSE (AI & ML) / CSE (DS) / CSE (CS) / IT								
Course Code	Category	Hours / Week			Credits	Maximum Marks		
ACSD02	Foundation	L	T	P	C	CIA	SEE	Total
		0	1	2	2	40	60	100
Contact Classes: 15	Tutorial Classes: NIL	Practical Classes: 30			Total Classes: 45			
Prerequisite: There are no prerequisites to take this course.								

I. COURSE OVERVIEW:

This course provides a solid foundation in object-oriented programming concepts and hands-on experience in using them. It introduces the concepts of abstraction and reusable code design via the object-oriented paradigm. Through a series of examples and exercises students gain coding skills and develop an understanding of professional programming practices. Mastering Java facilitate the learning of other technologies.

II. COURSES OBJECTIVES:

The students will try to learn

- I. The strong foundation with the Java Virtual Machine, its concepts and features.
- II. The systematic understanding of key aspects of the Java Class Library
- III. The usage of a modern IDE with an object oriented programming language to develop programs.

III. COURSE OUTCOMES:

At the end of the course students should be able to:

- CO 1 Develop non-trivial programs in an modern programming language.
- CO 2 Apply the principles of selection and iteration.
- CO 3 Appreciate uses of modular programming concepts for handling complex problems.
- CO 4 Recognize and apply principle features of object-oriented design such as abstraction and encapsulation.
- CO 5 Design classes with a view of flexibility and reusability.
- CO 6 Code, test and evaluate small use cases to conform to a specification.

IV. COURE CONTENT:

EXERCISES FOR OBJECT ORIENTED PROGRAMMING WITH JAVA LABORATORY

Note: Students are encouraged to bring their own laptops for laboratory practice sessions.

1. Getting Started Exercises

1.1 HelloWorld

1. Install JDK on your machine.
2. Write a Hello-world program using JDK and a source-code editor, such as:
 - For All Platforms: Sublime Text, Atom
 - For Windows: TextPad, NotePad++
 - For macOS: jEdit, gedit
 - For Ubuntu: gedit
3. Do ALL the exercises.

1.2 Writing Good Programs

The only way to learn programming is program, program and program. Learning programming is like learning cycling, swimming or any other sports. You can't learn by watching or reading books. Start to program immediately. On the other hands, to improve your programming, you need to read many books and study how the masters program.

It is easy to write programs that work. It is much harder to write programs that not only work but also easy to maintain and understood by others – I call these good programs. In the real world, writing program is not meaningful. You have to write good programs, so that others can understand and maintain your programs.

Pay particular attention to:

1. Coding Style:
 - Read "Java Code Convention" (@ <https://www.oracle.com/technetwork/java/codeconventions-150003.pdf> or google "Java Code Convention").
 - Follow the Java Naming Conventions for variables, methods, and classes STRICTLY. Use CamelCase for names. Variable and method names begin with lowercase, while class names begin with uppercase. Use nouns for variables (e.g., radius) and class names (e.g., Circle). Use verbs for methods (e.g., getArea(), isEmpty()).
 - **Use Meaningful Names:** Do not use names like a, b, c, d, x, x1, x2, and x1688 - they are meaningless. Avoid single-alphabet names like i, j, k. They are easy to type, but usually meaningless. Use single-alphabet names only when their meaning is clear, e.g., x, y, z for co-ordinates and i for array index. Use meaningful names like row and col (instead of x and y, i and j, x1 and x2), numStudents (not n), maxGrade, size (not n), and upperbound (not n again). Differentiate between singular and plural nouns (e.g., use books for an array of books, and book for each item).
 - Use consistent indentation and coding style. Many IDEs (such as Eclipse/NetBeans) can re-format your source codes with a single click.
2. Program Documentation: Comment! Comment! and more Comment to explain your code to other people and to yourself three days later.

1.3 Check Pass Fail (if-else)

Write a program called **CheckPassFail** which prints "PASS" if the int variable "mark" is more than or equal to 50; or prints "FAIL" otherwise. The program shall always print "DONE" before exiting.

Hints

Use >= for greater than or equal to comparison.

```
/* Trying if-else statement.
*/
public class CheckPassFail { // Save as "CheckPassFail.java"
    public static void main(String[] args) { // Program entry point
        int mark = 49; // Set the value of "mark" here!
        System.out.println("The mark is " + mark);
    }
}
```



```

// if-else statement
if ( ..... ) {
    System.out.println( ..... );
} else {
    System.out.println( ..... );
}
System.out.println( ..... );
}
}

```

Try

mark = 0, 49, 50, 51, 100 and verify your results.

Take note of the source-code **indentation!!!** Whenever you open a block with '{', indent all the statements inside the block by 3 (or 4 spaces). When the block ends, un-indent the closing '}' to align with the opening statement.

1.4 CheckOddEven (if-else)

Write a program called **CheckOddEven** which prints "Odd Number" if the int variable "number" is odd, or "Even Number" otherwise. The program shall always print "bye!" before exiting.

Hints

n is an even number if (n % 2) is 0; otherwise, it is an odd number. Use == for comparison, e.g., (n % 2) == 0.

```

/**
 * Trying if-else statement and modulus (%) operator.
 */
public class CheckOddEven { // Save as "CheckOddEven.java"
    public static void main(String[] args) { // Program entry point
        int number = 49; // Set the value of "number" here!
        System.out.println("The number is " + number);
        if ( ..... ) {
            System.out.println( ..... ); // even number
        } else {
            System.out.println( ..... ); // odd number
        }
        System.out.println( ..... );
    }
}

```

Try

number = 0, 1, 88, 99, -1, -2 and verify your results.

Again, take note of the source-code indentation! Make it a good habit to indent your code properly, for ease of reading your program.

1.5 PrintNumberInWord (nested-if, switch-case)

Write a program called **PrintNumberInWord** which prints "ONE", "TWO",... , "NINE", "OTHER" if the int variable "number" is 1, 2,... , 9, or other, respectively. Use (a) a "nested-if" statement; (b) a "switch-case-default" statement.

Hints

```

/**
 * Trying nested-if and switch-case statements.
 */
public class PrintNumberInWord { // Save as "PrintNumberInWord.java"
    public static void main(String[] args) {
        int number = 5; // Set the value of "number" here!
    }
}

```

```

// Using nested-if
if (number == 1) { // Use == for comparison
    System.out.println( ..... );
} else if ( ..... ) {
    .....
} else if ( ..... ) {
    .....
.....
.....
} else {
    .....
}

// Using switch-case-default
switch(number) {
    case 1:
        System.out.println( ..... ); break; // Don't forget the "break" after each case!
    case 2:
        System.out.println( ..... ); break;
    .....
    .....
    default: System.out.println( ..... );
}
}
}

```

Try

number = 0, 1, 2, 3, ..., 9, 10 and verify your results.

1.6 PrintDayInWord (nested-if, switch-case)

Write a program called **PrintDayInWord** which prints “Sunday”, “Monday”, ... “Saturday” if the int variable “dayNumber” is 0, 1, ..., 6, respectively. Otherwise, it shall print “Not a valid day”. Use (a) a “nested-if” statement; (b) a “switch-case-default” statement.

Try

dayNumber = 0, 1, 2, 3, 4, 5, 6, 7 and verify your results.

2. Exercises on Number Systems (for Science/Engineering Students)

To be proficient in programming, you need to be able to operate on these number systems:

1. Decimal (used by human beings for input and output)
2. Binary (used by computer for storage and processing)
3. Hexadecimal (shorthand or compact form for binary)

2.1 Exercises (Number Systems Conversion)

1. Convert the following decimal numbers into binary and hexadecimal numbers:
 - a. 108
 - b. 4848
 - c. 9000

Convert the following binary numbers into hexadecimal and decimal numbers:

- a. 10000000
- b. 101010101010
- c. 1000011000

Convert the following hexadecimal numbers into binary and decimal numbers:

- a. 1234
- b. 80F

c. ABCDE

Convert the following decimal numbers into binary equivalent:

- a. 123.456D
- b. 19.25D

2.2 Exercise (Integer Representation)

1. What are the ranges of 8-bit, 16-bit, 32-bit and 64-bit integer, in "unsigned" and "signed" representation?
2. Give the value of 88, 0, 1, 127, and 255 in 8-bit unsigned representation.
3. Give the value of +88, -88, -1, 0, +1, -128, and +127 in 8-bit 2's complement signed representation.
4. Give the value of +88, -88, -1, 0, +1, -127, and +127 in 8-bit sign-magnitude representation.
5. Give the value of +88, -88, -1, 0, +1, -127 and +127 in 8-bit 1's complement representation.

2.3 Exercises (Floating-point Numbers)

1. Compute the largest and smallest positive numbers that can be represented in the 32-bit normalized form.
2. Compute the largest and smallest negative numbers can be represented in the 32-bit normalized form.
3. Repeat (1) for the 32-bit denormalized form.
4. Repeat (2) for the 32-bit denormalized form.

Hints:

1. Largest positive number: S=0, E=1111 1110 (254), F=111 1111 1111 1111 1111 1111.
Smallest positive number: S=0, E=0000 0001 (1), F=000 0000 0000 0000 0000 0000.
2. Same as above, but S=1.
3. Largest positive number: S=0, E=0, F=111 1111 1111 1111 1111 1111.
Smallest positive number: S=0, E=0, F=000 0000 0000 0000 0000 0001.
4. Same as above, but S=1.

2.4 Exercises (Data Representation)

For the following 16-bit codes:

```
0000 0000 0010 1010;  
1000 0000 0010 1010;
```

Give their values, if they are representing:

1. a 16-bit unsigned integer;
2. a 16-bit signed integer;
3. two 8-bit unsigned integers;
4. two 8-bit signed integers;
5. a 16-bit Unicode characters;
6. two 8-bit ISO-8859-1 characters.

3. Exercises on Decision and Loop

3.1 SumAverageRunningInt (Decision & Loop)

Write a program called **SumAverageRunningInt** to produce the sum of 1, 2, 3, ..., to 100. Store 1 and 100 in variables lowerbound and upperbound, so that we can change their values easily. Also compute and display the average.

The output shall look like:

```
The sum of 1 to 100 is 5050  
The average is 50.5
```

Hints

```
/**
 * Compute the sum and average of running integers from a lowerbound to an upperbound using loop.
 */
public class SumAverageRunningInt { // Save as "SumAverageRunningInt.java"
    public static void main (String[] args) {
        // Define variables
        int sum = 0; // The accumulated sum, init to 0
        double average; // average in double
        final int LOWERBOUND = 1;
        final int UPPERBOUND = 100;

        // Use a for-loop to sum from lowerbound to upperbound
        for (int number = LOWERBOUND; number <= UPPERBOUND; ++number) {
            // The loop index variable number = 1, 2, 3, ..., 99, 100
            sum += number; // same as "sum = sum + number"
        }
        // Compute average in double. Beware that int / int produces int!
        .....
        // Print sum and average
        .....
    }
}
```

Try

1. Modify the program to use a "while-do" loop instead of "for" loop.

```
int sum = 0;
int number = LOWERBOUND; // declare and init loop index variable
while (number <= UPPERBOUND) { // test
    sum += number;
    ++number; // update
}
```

2. Modify the program using do-while loop

```
int sum = 0;
int number = LOWERBOUND; // declare and init loop index variable
do {
    sum += number;
    ++number; // update
} while (number <= UPPERBOUND); // test
```

3. What is the difference between "for" and "while-do" loops? What is the difference between "while-do" and "do-while" loops?
4. Modify the program to sum from 111 to 8899, and compute the average. Introduce an int variable called count to count the numbers in the specified range (to be used in computing the average).
5. Modify the program to find the "sum of the squares" of all the numbers from 1 to 100, i.e. $1*1 + 2*2 + 3*3 + \dots + 100*100$.
6. Modify the program to produce two sums: sum of odd numbers and sum of even numbers from 1 to 100. Also compute their absolute difference.

3.2 Product1ToN (or Factorial) (Decision & Loop)

Write a program called Product1ToN to compute the product of integers from 1 to 10 (i.e., $1 \times 2 \times 3 \times \dots \times 10$), as an int. Take note that It is the same as factorial of N.

Hints

Declare an int variable called product, initialize to 1, to accumulate the product.

```
// Define variables
int product = 1;    // The accumulated product, init to 1
final int LOWERBOUND = 1;
final int UPPERBOUND = 10;
```

Try

1. Compute the product from 1 to 11, 1 to 12, 1 to 13 and 1 to 14. Write down the product obtained and decide if the results are correct.

HINTS: Factorial of 13 (=6227020800) is outside the range of int [-2147483648, 2147483647]. Take note that computer programs may not produce the correct result even though the code seems correct!

2. Repeat the above, but use long to store the product. Compare the products obtained with int for N=13 and N=14.

HINTS: With long, you can store factorial of up to 20.

3.3 HarmonicSum (Decision & Loop)

Write a program called **HarmonicSum** to compute the sum of a harmonic series, as shown below, where n=50000. The program shall compute the sum from left-to-right as well as from the right-to-left. Are the two sums the same? Obtain the absolute difference between these two sums and explain the difference. Which sum is more accurate?

$$Harmonic(n) = 1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n}$$

Hints

```
/**
 * Compute the sum of harmonics series from left-to-right and right-to-left.
 */
public class HarmonicSum { // Save as "HarmonicSum.java"
    public static void main (String[] args) {
        // Define variables
        final int MAX_DENOMINATOR = 50000; // Use a more meaningful name instead of n
        double sumL2R = 0.0;    // Sum from left-to-right
        double sumR2L = 0.0;    // Sum from right-to-left
        double absDiff;        // Absolute difference between the two sums

        // for-loop for summing from left-to-right
        for (int denominator = 1; denominator <= MAX_DENOMINATOR; ++denominator) {
            // denominator = 1, 2, 3, 4, 5, ..., MAX_DENOMINATOR
            .....
            // Beware that int/int gives int, e.g., 1/2 gives 0.
        }
        System.out.println("The sum from left-to-right is: " + sumL2R);

        // for-loop for summing from right-to-left
        .....

        // Find the absolute difference and display
        if (sumL2R > sumR2L) .....
        else .....
    }
}
```

3.4 ComputePI (Decision & Loop)

Write a program called **ComputePI** to compute the value of π , using the following series expansion. Use the maximum denominator (MAX_DENOMINATOR) as the terminating condition. Try MAX_DENOMINATOR of 1000, 10000, 100000, 1000000 and compare the PI obtained. Is this series suitable for computing PI? Why?

$$\pi = 4 \times \left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + \frac{1}{13} - \frac{1}{15} + \dots \right)$$

Hints

Add to sum if the denominator % 4 is 1, and subtract from sum if it is 3.

```
double sum = 0.0;
int MAX_DENOMINATOR = 1000; // Try 10000, 100000, 1000000
for (int denominator = 1; denominator <= MAX_DENOMINATOR; denominator += 2) {
    // denominator = 1, 3, 5, 7, ..., MAX_DENOMINATOR
    if (denominator % 4 == 1) {
        sum += .....;
    } else if (denominator % 4 == 3) {
        sum -= .....;
    } else { // remainder of 0 or 2
        System.out.println("Impossible!!!");
    }
}
.....
```

Try

1. Instead of using maximum denominator as the terminating condition, rewrite your program to use the maximum number of terms (MAX_TERM) as the terminating condition.

```
int MAX_TERM = 10000; // number of terms used in computation
int sum = 0.0;
for (int term = 1; term <= MAX_TERM; term++) {
    // term = 1, 2, 3, 4, ..., MAX_TERM
    if (term % 2 == 1) { // odd term number: add
        sum += 1.0 / (term * 2 - 1);
    } else { // even term number: subtract
        .....
    }
}
}
```

2. JDK maintains the value of π in a built-in double constant called Math.PI (=3.141592653589793). Add a statement to compare the values obtained and the Math.PI, in percents of Math.PI, i.e., (piComputed / Math.PI) * 100.

3.5 CozaLozaWoza (Decision & Loop)

Write a program called **CozaLozaWoza** which prints the numbers 1 to 110, 11 numbers per line. The program shall print "Coza" in place of the numbers which are multiples of 3, "Loza" for multiples of 5, "Woza" for multiples of 7, "CozaLoza" for multiples of 3 and 5, and so on. The output shall look like:

```
1 2 Coza 4 Loza Coza Woza 8 Coza Loza 11
Coza 13 Woza CozaLoza 16 17 Coza 19 Loza CozaWoza 22
23 Coza Loza 26 Coza Woza 29 CozaLoza 31 32 Coza
.....
```

Hints

```
public class CozaLozaWoza { // Save as "CozaLozaWoza.java"
    public static void main(String[] args) {
        final int LOWERBOUND = 1, UPPERBOUND = 110;
        for (int number = LOWERBOUND; number <= UPPERBOUND; ++number) {
            // number = LOWERBOUND+1, LOWERBOUND+2, ..., UPPERBOUND
            // Print "Coza" if number is divisible by 3
            .....
        }
    }
}
```

```

if ( ..... ) {
    System.out.print("Coza");
}
// Print "Loza" if number is divisible by 5
if ( ..... ) {
    System.out.print(.....);
}
// Print "Woza" if number is divisible by 7
.....
// Print the number if it is not divisible by 3, 5 and 7 (i.e., it has not been processed above)
if ( ..... ) {
    .....
}
// After processing the number, print a newline if number is divisible by 11;
// else print a space
if ( ..... ) {
    System.out.println(); // print newline
} else {
    System.out.print( ..... ); // print a space
}
}
}
}

```

Notes

1. You cannot use nested-if (if ... else if ... else if ... else) for this problem. It is because the tests are not mutually exclusive. For example, 15 is divisible by both 3 and 5. Nested-if is only applicable if the tests are mutually exclusive.
2. The tests above look messy. A better solution is to use a boolean flag to keep track of whether the number has been processed, as follows:

```

final int LOWERBOUND = 1, UPPERBOUND = 110;
boolean printed;
for (int number = LOWERBOUND; number <= UPPERBOUND; ++number) {
    printed = false; // init before processing each number
    // Print "Coza" if number is divisible by 3
    if ( ..... ) {
        System.out.print( ..... );
        printed = true; // processed!
    }
    // Print "Loza" if number is divisible by 5
    if ( ..... ) {
        System.out.print( ..... );
        printed = true; // processed!
    }
    // Print "Woza" if number is divisible by 7
    .....
    // Print the number if it has not been processed
    if (!printed) {
        .....
    }
    // After processing the number, print a newline if it is divisible by 11;
    // else, print a space
    .....
}
}

```


3.6 Fibonacci (Decision & Loop)

Write a program called **Fibonacci** to print the first 20 Fibonacci numbers $F(n)$, where $F(n)=F(n-1)+F(n-2)$ and $F(1)=F(2)=1$. Also compute their average. The output shall look like:

The first 20 Fibonacci numbers are:

1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765

The average is 885.5

Hints

```
/**
 * Print first 20 Fibonacci numbers and their average
 */
public class Fibonacci {
    public static void main (String[] args) {
        int n = 3;        // The index n for F(n), starting from n=3, as n=1 and n=2 are pre-defined
        int fn;           // F(n) to be computed
        int fnMinus1 = 1; // F(n-1), init to F(2)
        int fnMinus2 = 1; // F(n-2), init to F(1)
        int nMax = 20;    // maximum n, inclusive
        int sum = fnMinus1 + fnMinus2; // Need sum to compute average
        double average;

        System.out.println("The first " + nMax + " Fibonacci numbers are:");
        .....

        while (n <= nMax) { // n starts from 3
            // n = 3, 4, 5, ..., nMax
            // Compute F(n), print it and add to sum
            .....
            // Increment the index n and shift the numbers for the next iteration
            ++n;
            fnMinus2 = fnMinus1;
            fnMinus1 = fn;
        }

        // Compute and display the average (=sum/nMax).
        // Beware that int/int gives int.
        .....
    }
}
```

Try

1. Tribonacci numbers are a sequence of numbers $T(n)$ similar to Fibonacci numbers, except that a number is formed by adding the three previous numbers, i.e., $T(n)=T(n-1)+T(n-2)+T(n-3)$, $T(1)=T(2)=1$, and $T(3)=2$. Write a program called **Tribonacci** to produce the first twenty Tribonacci numbers.

3.7 ExtractDigits (Decision & Loop)

Write a program called **ExtractDigits** to extract each digit from an int, in the reverse order. For example, if the int is 15423, the output shall be "3 2 4 5 1", with a space separating the digits.

Hints

The coding pattern for extracting individual digits from an integer n is:

1. Use $(n \% 10)$ to extract the last (least-significant) digit.
2. Use $n = n / 10$ to drop the last (least-significant) digit.
3. Repeat if $(n > 0)$, i.e., more digits to extract.

Take note that n is destroyed in the process. You may need to clone a copy.

```
int n = ...;
while (n > 0) {
    int digit = n % 10; // Extract the least-significant digit
    // Print this digit
    .....
    n = n / 10; // Drop the least-significant digit and repeat the loop
}
```

4. Exercises on Input, Decision and Loop

4.1 Add2Integer (Input)

Write a program called Add2Integers that prompts user to enter two integers. The program shall read the two integers as int; compute their sum; and print the result. For example,

```
Enter first integer: 8
Enter second integer: 9
The sum is: 17
```

Hints

```
import java.util.Scanner; // For keyboard input
/**
 * 1. Prompt user for 2 integers
 * 2. Read inputs as "int"
 * 3. Compute their sum in "int"
 * 4. Print the result
 */
public class Add2Integers { // Save as "Add2Integers.java"
    public static void main (String[] args) {
        // Declare variables
        int number1, number2, sum;

        // Put up prompting messages and read inputs as "int"
        Scanner in = new Scanner(System.in); // Scan the keyboard for input
        System.out.print("Enter first integer: "); // No newline for prompting message
        number1 = in.nextInt(); // Read next input as "int"
        .....
        in.close(); // Close Scanner

        // Compute sum
        sum = .....

        // Display result
        System.out.println("The sum is: " + sum); // Print with newline
    }
}
```

4.2 SumProductMinMax3 (Arithmetic & Min/Max)

Write a program called SumProductMinMax3 that prompts user for three integers. The program shall read the inputs as int; compute the sum, product, minimum and maximum of the three integers; and print the results. For example,

```
Enter 1st integer: 8
Enter 2nd integer: 2
Enter 3rd integer: 9
The sum is: 19
The product is: 144
The min is: 2
```

The max is: 9

Hints

```
// Declare variables
int number1, number2, number3; // The 3 input integers
int sum, product, min, max;    // To compute these

// Prompt and read inputs as "int"
Scanner in = new Scanner(System.in); // Scan the keyboard
.....
.....
in.close();

// Compute sum and product
sum = .....
product = .....

// Compute min
// The "coding pattern" for computing min is:
// 1. Set min to the first item
// 2. Compare current min with the second item and update min if second item is smaller
// 3. Repeat for the next item
min = number1;    // Assume min is the 1st item
if (number2 < min) { // Check if the 2nd item is smaller than current min
    min = number2; // Update min if so
}
if (number3 < min) { // Continue for the next item
    min = number3;
}

// Compute max - similar to min
.....

// Print results
.....
```

Try

1. Write a program called **SumProductMinMax5** that prompts user for five integers. The program shall read the inputs as int; compute the sum, product, minimum and maximum of the five integers; and print the results. Use five int variables: number1, number2, ..., number5 to store the inputs.

4.3 CircleComputation (double & printf())

Write a program called **CircleComputation** that prompts user for the radius of a circle in floating point number. The program shall read the input as double; compute the diameter, circumference, and area of the circle in double; and print the values rounded to 2 decimal places. Use System-provided constant Math.PI for pi. The formulas are:

```
diameter = 2.0 * radius;
area = Math.PI * radius * radius;
circumference = 2.0 * Math.PI * radius;
```

Hints

```
// Declare variables
double radius, diameter, circumference, area; // inputs and results - all in double
.....

// Prompt and read inputs as "double"
System.out.print("Enter the radius: ");
```

```
radius = in.nextDouble(); // read input as double

// Compute in "double"
.....

// Print results using printf() with the following format specifiers:
// %.2f for a double with 2 decimal digits
// %n for a newline
System.out.printf("Diameter is: %.2f%n", diameter);
.....
```

Try

1. Write a program called **SphereComputation** that prompts user for the radius of a sphere in floating point number. The program shall read the input as double; compute the volume and surface area of the sphere in double; and print the values rounded to 2 decimal places. The formulas are:

```
surfaceArea = 4 * Math.PI * radius * radius;
volume = 4 / 3 * Math.PI * radius * radius * radius; // But this does not work in programming?! Why?
```

Take note that you cannot name the variable surface area with a space or surface-area with a dash. Java's naming convention is surfaceArea. Other languages recommend surface_area with an underscore.

2. Write a program called **CylinderComputation** that prompts user for the base radius and height of a cylinder in floating point number. The program shall read the inputs as double; compute the base area, surface area, and volume of the cylinder; and print the values rounded to 2 decimal places. The formulas are:

```
baseArea = Math.PI * radius * radius;
surfaceArea = 2.0 * Math.PI * radius + 2.0 * baseArea;
volume = baseArea * height;
```

4.4 Swap2Integers

Write a program called Swap2Integers that prompts user for two integers. The program shall read the inputs as int, save in two variables called number1 and number2; swap the contents of the two variables; and print the results. For examples,

```
Enter first integer: 9
Enter second integer: -9
After the swap, first integer is: -9, second integer is: 9
```

Hints

To swap the contents of two variables x and y, you need to introduce a temporary storage, say temp, and do: temp \leftarrow x; x \leftarrow y; y \leftarrow temp.

4.5 IncomeTaxCalculator (Decision)

The progressive income tax rate is mandated as follows:

Taxable Income	Rate (%)
First \$20,000	0
Next \$20,000	10
Next \$20,000	20
The remaining	30

For example, suppose that the taxable income is \$85000, the income tax payable is $\$20000 \times 0\% + \$20000 \times 10\% + \$20000 \times 20\% + \$25000 \times 30\%$.

Write a program called **IncomeTaxCalculator** that reads the taxable income (in int). The program shall calculate the income tax payable (in double); and print the result rounded to 2 decimal places. For examples,

Enter the taxable income: **\$41234**
The income tax payable is: \$2246.80

Enter the taxable income: **\$67891**
The income tax payable is: \$8367.30

Enter the taxable income: **\$85432**
The income tax payable is: \$13629.60

Enter the taxable income: **\$12345**
The income tax payable is: \$0.00

Hints

```
// Declare constants first (variables may use these constants)
// The keyword "final" marked these as constant (i.e., cannot be changed).
// Use uppercase words joined with underscore to name constants
final double TAX_RATE_ABOVE_20K = 0.1;
final double TAX_RATE_ABOVE_40K = 0.2;
final double TAX_RATE_ABOVE_60K = 0.3;

// Declare variables
int taxableIncome;
double taxPayable;
.....

// Compute tax payable in "double" using a nested-if to handle 4 cases
if (taxableIncome <= 20000) { // [0, 20000]
    taxPayable = .....;
} else if (taxableIncome <= 40000) { // [20001, 40000]
    taxPayable = .....;
} else if (taxableIncome <= 60000) { // [40001, 60000]
    taxPayable = .....;
} else { // [60001, ]
    taxPayable = .....;
}

// Alternatively, you could use the following nested-if conditions
// but the above follows the table data
//if (taxableIncome > 60000) { // [60001, ]
//    .....
//} else if (taxableIncome > 40000) { // [40001, 60000]
//    .....
//} else if (taxableIncome > 20000) { // [20001, 40000]
//    .....
//} else { // [0, 20000]
//    .....
//}

// Print results rounded to 2 decimal places
System.out.printf("The income tax payable is: %.2f%n", ...);
```

Try

Suppose that a 10% tax rebate is announced for the income tax payable, capped at \$1,000, modify your program to handle the tax rebate. For example, suppose that the tax payable is \$12,000, the rebate is \$1,000, as 10% of \$12,000 exceed the cap.

4.6 IncomeTaxCalculatorWithSentinel (Decision & Loop)

Based on the previous exercise, write a program called `IncomeTaxCalculatorWithSentinel` which shall repeat the calculation until user enter -1. For example,

Enter the taxable income (or -1 to end): **\$41000**

The income tax payable is: \$2200.00

Enter the taxable income (or -1 to end): **\$62000**

The income tax payable is: \$6600.00

Enter the taxable income (or -1 to end): **\$73123**

The income tax payable is: \$9936.90

Enter the taxable income (or -1 to end): **\$84328**

The income tax payable is: \$13298.40

Enter the taxable income: **\$-1**

bye!

The -1 is known as the sentinel value. (Wiki: In programming, a sentinel value, also referred to as a flag value, trip value, rogue value, signal value, or dummy data, is a special value which uses its presence as a condition of termination.)

Hints

The coding pattern for handling input with sentinel value is as follows:

```
// Declare constants first
final int SENTINEL = -1; // Terminating value for input
.....
// Declare variables
int taxableIncome;
double taxPayable;
.....

// Read the first input to "seed" the while loop
System.out.print("Enter the taxable income (or -1 to end): $");
taxableIncome = in.nextInt();

while (taxableIncome != SENTINEL) {
    // Compute tax payable
    .....
    // Print result
    .....

    // Read the next input
    System.out.print("Enter the taxable income (or -1 to end): $");
    taxableIncome = in.nextInt();
    // Repeat the loop body, only if the input is not the SENTINEL value.
    // Take note that you need to repeat these two statements inside/outside the loop!
}
System.out.println("bye!");
```

Take note that we repeat the input statements inside and outside the loop. Repeating statements is NOT a good programming practice. This is because it is easy to repeat (Cntl-C/Cntl-V), but hard to maintain and synchronize the repeated statements. In this case, we have no better choices!

4.7 PensionContributionCalculatorWithSentinel (Decision & Loop)

Based on the previous PensionContributionCalculator, write a program called **PensionContributionCalculatorWithSentinel** which shall repeat the calculations until user enter -1 for the salary. For examples,

Enter the monthly salary (or -1 to end): **\$5123**

Enter the age: **21**

The employee's contribution is: \$1024.60

The employer's contribution is: \$870.91

The total contribution is: \$1895.51

Enter the monthly salary (or -1 to end): **\$5123**
Enter the age: **64**
The employee's contribution is: \$384.22
The employer's contribution is: \$461.07
The total contribution is: \$845.30

Enter the monthly salary (or -1 to end): **\$-1**
bye!

Hints

```
// Read the first input to "seed" the while loop
System.out.print("Enter the monthly salary (or -1 to end): $");
salary = in.nextInt();

while (salary != SENTINEL) {
    // Read the remaining
    System.out.print("Enter the age: ");
    age = in.nextInt();

    .....

    .....

    // Read the next input and repeat
    System.out.print("Enter the monthly salary (or -1 to end): $");
    salary = in.nextInt();
}
```

4.8 SalesTaxCalculator (Decision & Loop)

A sales tax of 7% is levied on all goods and services consumed. It is also mandatory that all the price tags should include the sales tax. For example, if an item has a price tag of \$107, the actual price is \$100 and \$7 goes to the sales tax.

Write a program using a loop to continuously input the tax-inclusive price (in double); compute the actual price and the sales tax (in double); and print the results rounded to 2 decimal places. The program shall terminate in response to input of -1; and print the total price, total actual price, and total sales tax. For examples,

Enter the tax-inclusive price in dollars (or -1 to end): **107**
Actual Price is: \$100.00, Sales Tax is: \$7.00

Enter the tax-inclusive price in dollars (or -1 to end): **214**
Actual Price is: \$200.00, Sales Tax is: \$14.00

Enter the tax-inclusive price in dollars (or -1 to end): **321**
Actual Price is: \$300.00, Sales Tax is: \$21.00

Enter the tax-inclusive price in dollars (or -1 to end): **-1**
Total Price is: \$642.00
Total Actual Price is: \$600.00
Total Sales Tax is: \$42.00

Hints

```
// Declare constants
final double SALES_TAX_RATE = 0.07;
final int SENTINEL = -1;    // Terminating value for input

// Declare variables
double price, actualPrice, salesTax; // inputs and results
double totalPrice = 0.0, totalActualPrice = 0.0, totalSalesTax = 0.0; // to accumulate
.....
```

```
// Read the first input to "seed" the while loop
System.out.print("Enter the tax-inclusive price in dollars (or -1 to end): ");
price = in.nextDouble();

while (price != SENTINEL) {
    // Compute the tax
    .....
    // Accumulate into the totals
    .....
    // Print results
    .....

    // Read the next input and repeat
    System.out.print("Enter the tax-inclusive price in dollars (or -1 to end): ");
    price = in.nextDouble();
}
// print totals
.....
```

4.9 ReverseInt (Loop with Modulus/Divide)

Write a program that prompts user for a positive integer. The program shall read the input as int; and print the "reverse" of the input integer. For examples,

Enter a positive integer: **12345**
The reverse is: 54321

Hints

Use the following coding pattern which uses a while-loop with repeated modulus/divide operations to extract and drop the last digit of a positive integer.

```
// Declare variables
int inNumber; // to be input
int inDigit; // each digit
.....

// Extract and drop the "last" digit repeatably using a while-loop with modulus/divide operations
while (inNumber > 0) {
    inDigit = inNumber % 10; // extract the "last" digit
    // Print this digit (which is extracted in reverse order)
    .....
    inNumber /= 10; // drop "last" digit and repeat
}
.....
```

4.10 SumOfDigitsInt (Loop with Modulus/Divide)

Write a program that prompts user for a positive integer. The program shall read the input as int; compute and print the sum of all its digits. For examples,

Enter a positive integer: **12345**
The sum of all digits is: 15

Hints

See "ReverseInt".

4.11 InputValidation (Loop with boolean flag)

Your program often needs to validate the user's inputs, e.g., marks shall be between 0 and 100.

Write a program that prompts user for an integer between 0-10 or 90-100. The program shall read the input as int; and repeat until the user enters a valid input. For examples,

```
Enter a number between 0-10 or 90-100: -1
Invalid input, try again...
Enter a number between 0-10 or 90-100: 50
Invalid input, try again...
Enter a number between 0-10 or 90-100: 101
Invalid input, try again...
Enter a number between 0-10 or 90-100: 95
You have entered: 95
```

Hints

Use the following coding pattern which uses a do-while loop controlled by a boolean flag to do input validation. We use a do-while instead of while-do loop as we need to execute the body to prompt and process the input at least once.

```
// Declare variables
int numberIn;    // to be input
boolean isValid; // boolean flag to control the loop
.....

// Use a do-while loop controlled by a boolean flag
// to repeatably read the input until a valid input is entered
isValid = false; // default assuming input is not valid
do {
    // Prompt and read input
    .....

    // Validate input by setting the boolean flag accordingly
    if (numberIn ..... ) {
        isValid = true; // exit the loop
    } else {
        System.out.println(.....); // Print error message and repeat
    }
} while (!isValid);
.....
```

4.12 AverageWithInputValidation (Loop with boolean flag)

Write a program that prompts user for the mark (between 0-100 in int) of 3 students; computes the average (in double); and prints the result rounded to 2 decimal places. Your program needs to perform input validation. For examples,

```
Enter the mark (0-100) for student 1: 56
Enter the mark (0-100) for student 2: 101
Invalid input, try again...
Enter the mark (0-100) for student 2: -1
Invalid input, try again...
Enter the mark (0-100) for student 2: 99
Enter the mark (0-100) for student 3: 45
The average is: 66.67
```

Hints

```

// Declare constant
final int NUM_STUDENTS = 3;

// Declare variables
int numberIn;
boolean isValid; // boolean flag to control the input validation loop
int sum = 0;
double average;
.....

for (int studentNo = 1; studentNo <= NUM_STUDENTS; ++studentNo) {
    // Prompt user for mark with input validation
    .....
    isValid = false; // reset assuming input is not valid
    do {
        .....
    } while (!isValid);

    sum += .....;
}
.....

```

5. Exercises on Nested-Loops

5.1 SquarePattern (nested-loop)

Write a program called **SquarePattern** that prompts user for the size (a non-negative integer in int); and prints the following square pattern using two nested for-loops.

Enter the size: 5

```

#####
#####
#####
#####
#####
#####

```

Hints

The code pattern for printing 2D patterns using nested loops is:

```

// Outer loop to print each of the rows
for (int row = 1; row <= size; row++) { // row = 1, 2, 3, ..., size
    // Inner loop to print each of the columns of a particular row
    for (int col = 1; col <= size; col++) { // col = 1, 2, 3, ..., size
        System.out.print( ..... ); // Use print() without newline inside the inner loop
        .....
    }
    // Print a newline after printing all the columns
    System.out.println();
}

```

Notes

1. You should name the loop indexes row and col, NOT i and j, or x and y, or a and b, which are meaningless.
2. The row and col could start at 1 (and upto size), or start at 0 (and upto size-1). As computer counts from 0, it is probably more efficient to start from 0. However, since humans counts from 1, it is easier to read if you start from 1.

Try

Rewrite the above program using nested while-do loops.

5.2 CheckerPattern (nested-loop)

Write a program called **CheckerPattern** that prompts user for the size (a non-negative integer in int); and prints the following checkerboard pattern.

```
Enter the size: 7
# # # # #
# # # # #
# # # # #
# # # # #
# # # # #
# # # # #
# # # # #
```

Hints

```
// Outer loop to print each of the rows
for (int row = 1; row <= size; row++) { // row = 1, 2, 3, ..., size
    // Inner loop to print each of the columns of a particular row
    for (int col = 1; col <= size; col++) { // col = 1, 2, 3, ..., size
        if ((row % 2) == 0) { // row 2, 4, 6, ...
            .....
        }
        System.out.print( ..... ); // Use print() without newline inside the inner loop
        .....
    }
    // Print a newline after printing all the columns
    System.out.println();
}
```

5.3 TimeTable (nested-loop)

Write a program called **TimeTable** that prompts user for the size (a positive integer in int); and prints the multiplication table as shown:

```
Enter the size: 10
*| 1 2 3 4 5 6 7 8 9 10
-----
1| 1 2 3 4 5 6 7 8 9 10
2| 2 4 6 8 10 12 14 16 18 20
3| 3 6 9 12 15 18 21 24 27 30
4| 4 8 12 16 20 24 28 32 36 40
5| 5 10 15 20 25 30 35 40 45 50
6| 6 12 18 24 30 36 42 48 54 60
7| 7 14 21 28 35 42 49 56 63 70
8| 8 16 24 32 40 48 56 64 72 80
9| 9 18 27 36 45 54 63 72 81 90
10| 10 20 30 40 50 60 70 80 90 100
```

Hints

1. Use printf() to format the output, e.g., each cell is %4d.
2. See "Java Basics" article.

5.4 TriangularPattern (nested-loop)

Write 4 programs called **TriangularPatternX** (X = A, B, C, D) that prompts user for the size (a non-negative integer in int); and prints each of the patterns as shown:

```
Enter the size: 8
```

```

#           #####          #####          #
##          #####          #####          ##
###         #####          #####          ###
####        #####          #####          ####
#####       #####          #####          #####
#####      #####          #####          #####
#####     #####          #####          #####
#####    #####          #####          #####
#####   #####          #####          #####
#####  #####          #####          #####
##### #####          #####          #####
(a)      (b)      (c)      (d)

```

Hints

1. On the main diagonal, $row = col$. On the opposite diagonal, $row + col = size + 1$, where row and col begin from 1.
2. You need to print the leading blanks, in order to push the # to the right. The trailing blanks are optional, which does not affect the pattern.
3. For pattern (a), if $(row \geq col)$ print #. Trailing blanks are optional.
4. For pattern (b), if $(row + col \leq size + 1)$ print #. Trailing blanks are optional.
5. For pattern (c), if $(row \geq col)$ print #; else print blank. Need to print the leading blanks.
6. For pattern (d), if $(row + col \geq size + 1)$ print #; else print blank. Need to print the leading blanks.
7. The coding pattern is:

```

// Outer loop to print each of the rows
for (int row = 1; row <= size; row++) { // row = 1, 2, 3, ..., size
    // Inner loop to print each of the columns of a particular row
    for (int col = 1; col <= size; col++) { // col = 1, 2, 3, ..., size
        if (.....) {
            System.out.print("# ");
        } else {
            System.out.print(" "); // Need to print the "leading" blanks
        }
    }
    // Print a newline after printing all the columns
    System.out.println();
}

```

5.5 BoxPattern (nested-loop)

Write 4 programs called **BoxPatternX** ($X = A, B, C, D$) that prompts user for the size (a non-negative integer in int); and prints the pattern as shown:

Enter the size: 8

```

#####          #####          #####          #####          #####
#      #      #          #      #      #      #      #
#      #      #          #      #      #      #      #
#      #      #          #      #      #      #      #
#      #      #          #      #      #      #      #
#      #      #          #      #      #      #      #
#####          #####          #####          #####          #####
(a)      (b)      (c)      (d)      (e)

```

Hints

1. On the main diagonal, $row = col$. On the opposite diagonal, $row + col = size + 1$, where row and col begin from 1.
2. For pattern (a), if $(row == 1 \parallel row == size \parallel col == 1 \parallel col == size)$ print #; else print blank. Need to print the intermediate blanks.
3. For pattern (b), if $(row == 1 \parallel row == size \parallel row == col)$ print #; else print blank.

5.6 HillPattern (nested-loop)

Write 3 programs called **HillPatternX** (X = A, B, C, D) that prompts user for the size (a non-negative integer in int); and prints the pattern as shown:

Enter the rows: 6

```

      #      #####
    ###      #####
  #####      #####
#####      #####
#####      #####
#####      #####
(a)          (b)
#####      #####
#####      #####
#####      #####
#####      #####
#          #####
(c)          (d)

```

Hints

For pattern (a):

```

for (int row = 1; ..... ) {
    // numCol = 2*numRows - 1
    for (int col = 1; ..... ) {
        if ((row + col >= numRows + 1) && (row >= col - numRows + 1)) {
            .....;
        } else {
            .....;
        }
    }
    .....;
}

```

or, use 2 sequential inner loops to print the columns:

```

for (int row = 1; row <= rows; row++) {
    for (int col = 1; col <= rows; col++) {
        if ((row + col >= rows + 1)) {
            .....
        } else {
            .....
        }
    }
    for (int col = 2; col <= rows; col++) { // skip col = 1
        if (row >= col) {
            .....
        } else {
            .....
        }
    }
    .....
}

```

5.7 NumberPattern (nested-loop)

Write 4 programs called **NumberPatternX** (X = A, B, C, D) that prompts user for the size (a non-negative

integer in int); and prints the pattern as shown:

Enter the size: 8

1	1 2 3 4 5 6 7 8	1	8 7 6 5 4 3 2 1
1 2	1 2 3 4 5 6 7	2 1	7 6 5 4 3 2 1
1 2 3	1 2 3 4 5 6	3 2 1	6 5 4 3 2 1
1 2 3 4	1 2 3 4 5	4 3 2 1	5 4 3 2 1
1 2 3 4 5	1 2 3 4	5 4 3 2 1	4 3 2 1
1 2 3 4 5 6	1 2 3	6 5 4 3 2 1	3 2 1
1 2 3 4 5 6 7	1 2	7 6 5 4 3 2 1	2 1
1 2 3 4 5 6 7 8	1	8 7 6 5 4 3 2 1	1
(a)	(b)	(c)	(d)

6. Magic(Special) Numbers

6.1. Amicable umbers

Two different numbers are said to be so Amicable numbers if each sum of divisors is equal to the other number. Amicable Numbers are: (220, 284), (1184, 1210), (2620, 2924), (5020, 5564), (6232, 6368). For example,

Enter 1st number: 228

Enter 2nd number: 220

The numbers are Amicable Numbers.

Hints

220 and 284 are Amicable Numbers.

Divisors of 220 = 1, 2, 4, 5, 10, 11, 20, 22, 44, 55, 110

$1+2+4+5+10+11+20+22+44+55+110 = 284$

Divisors of 284 = 1, 2, 4, 71, 142

$1+2+4+71+142 = 220$

6.2. Armstrong Number

Armstrong number is a positive number if it is equal to the sum of cubes of its digits is called Armstrong number and if its sum is not equal to the number then it's not a Armstrong number. For example,

Enter number=145

145 is not an Armstrong Number

Enter number = 153

153 is an Armstrong Number

Hints

Examples: 153 is Armstrong

$(1*1*1)+(5*5*5)+(3*3*3) = 153$

6.3. Capricorn Number

A number is called Capricorn or Kaprekar number whose square is divided into two parts in any conditions and parts are added, the additions of parts is equal to the number, is called Capricorn or Kaprekar number. For example,

```
Enter a number : 45
45 is a Capricorn/Kaprekar number
Enter a number : 297
297 is a Capricorn/Kaprekar number
Enter a number : 44
44 is not a Capricorn/Kaprekar number
```

Hints

```
Number = 45
(45)2 = 2025

All parts for 2025:
202 + 5 = 207 (not 45)
20 + 25 = 45
2 + 025 = 27 (not 45)
```

From the above we can see one combination is equal to number so that 45 is Capricorn or Kaprekar number.

Try

Write a Java program to generate and show all Kaprekar numbers less than 1000.

6.4. Circular Prime

A circular prime is a prime number with the property that the number generated at each intermediate step when cyclically permuting its digits will be prime. For example, 1193 is a circular prime, since 1931, 9311 and 3119 all are also prime. For example,

```
Enter a number : 137
137 is a Circular Prime
Enter a number : 44
44 is not a Circular Prime
```

6.5. Happy Number

A happy number is a natural number in a given number base that eventually reaches 1 when iterated over the perfect digital invariant function for. Those numbers that do not end in 1 are -unhappy numbers. For example,

```
Enter a number : 31
31 is a Happy number

Enter a number : 32
32 is not a Happy number
```

6.6. Automorphic Number

An Automorphic number is a number whose square “ends” in the same digits as the number itself. For example,

```
Enter a number : 5
5 is a Automorphic Number

Enter a number : 25
25 is a Automorphic Number

Enter a number : 2
2 is not a Automorphic Number
```

Hints

```
5*5 = 25, 6*6 = 36, 25*25 = 625

5,6,25 are automorphic numbers
```

6.7. Disarium Number

A number is called Disarium number if the sum of its power of the positions from left to right is equal to the number. For example,

Enter a number : 135
135 is a Disarium Number

Enter a number : 32
32 is not a Disarium Number

Hints

$$1^1 + 3^2 + 5^3 = 1 + 9 + 125 = 135$$

6.8. Magic Number

Magic number is the if the sum of its digits recursively are calculated till a single digit. If the single digit is 1 then the number is a magic number. Magic number is very similar with Happy Number. For example,

Enter a number : 226
226 is a Magic Number

Enter a number : 32
32 is not a Magic Number
Enter number = 541
153 is a Magic Number

Hints

226 is said to be a magic number

$2+2+6=10$ sum of digits is 10 then again $1+0=1$ now we get a single digit number is 1. If we single digit number will now 1 then it would not be a magic number.

6.9. Neon Number

A neon number is a number where the sum of digits of square of the number is equal to the number. For example if the input number is 9, its square is $9*9 = 81$ and sum of the digits is 9. i.e. 9 is a neon number. For example,

Enter a number: 9
9 is a Neon Number

Enter a number: 8
8 is not a Neon Number

6.10. Palindromic Number

A palindromic number is a number that remains the same when its digits are reversed. For example,

Enter a number : 16461
16461 is a Palindromic Number

Enter a number : 1234
1234 is not a Palindromic Number

6.11. Perfect Number

A perfect number is a positive integer that is equal to the sum of its positive divisors, excluding the number itself. For instance, 6 has divisors 1, 2 and 3, and $1 + 2 + 3 = 6$, so 6 is a perfect number. For example,

Enter a number : 6
6 is a Perfect Number

Enter a number : 3
3 is not a Perfect Number

6.12. Special Number

A number is said to be special number when the sum of factorial of its digits is equal to the number itself. Example- 145 is a Special Number as $1!+4!+5!=145$. For example,

Enter a number : 145
145 is a Special Number

Enter a number : 23
23 is not a Special Number

6.13. Spy Number

A spy number is a number where the sum of its digits equals the product of its digits. For example, 1124 is a spy number, the sum of its digits is $1+1+2+4=8$ and the product of its digits is $1*1*2*4=8$. For example,

Enter a number : 1124
1124 is a Spy Number

Enter a number : 12
12 is not a Spy Number

6.14. Ugly Number

A number is said to be an Ugly number if positive numbers whose prime factors only include 2, 3, 5. For example, $6(2 \times 3)$, $8(2 \times 2 \times 2)$, $15(3 \times 5)$ are ugly numbers while $14(2 \times 7)$ is not ugly since it includes another prime factor 7. Note that 1 is typically treated as an ugly number. For example,

Enter a number : 6
6 is an Ugly Number

Enter a number : 14
14 is not an Ugly Number

7. Exercises on String and char Operations

7.1 ReverseString (String & char)

Write a program called **ReverseString**, which prompts user for a String, and prints the reverse of the String by extracting and processing each character. The output shall look like:

Enter a String: **abcdef**
The reverse of the String "abcdef" is "fedcba".

Hints

For a String called inStr, you can use inStr.length() to get the length of the String; and inStr.charAt(idx) to retrieve the char at the idx position, where idx begins at 0, up to inStr.length() - 1.

```
// Define variables
String inStr;    // input String
int inStrLen;    // length of the input String
.....

// Prompt and read input as "String"
System.out.print("Enter a String: ");
inStr = in.next(); // use next() to read a String
inStrLen = inStr.length();

// Use inStr.charAt(index) in a loop to extract each character
```

```
// The String's index begins at 0 from the left.
// Process the String from the right
for (int charIdx = inStrLen - 1; charIdx >= 0; --charIdx) {
    // charIdx = inStrLen-1, inStrLen-2, ... ,0
    .....
}
```

7.2 CountVowelsDigits (String & char)

Write a program called **CountVowelsDigits**, which prompts the user for a String, counts the number of vowels (a, e, i, o, u, A, E, I, O, U) and digits (0-9) contained in the string, and prints the counts and the percentages (rounded to 2 decimal places). For example,

```
Enter a String: testing12345
Number of vowels: 2 (16.67%)
Number of digits: 5 (41.67%)
```

Hints

1. To check if a char c is a digit, you can use boolean expression (c >= '0' && c <= '9'); or use built-in boolean function Character.isDigit(c).
2. You could use in.next().toLowerCase() to convert the input String to lowercase to reduce the number of cases.
3. To print a % using printf(), you need to use %%. This is because % is a prefix for format specifier in printf(), e.g., %d and %f.

7.3 PhoneKeyPad (String & char)

On you phone keypad, the alphabets are mapped to digits as follows:

ABC(2), DEF(3), GHI(4), JKL(5), MNO(6), PQRS(7), TUV(8), WXYZ(9).

Write a program called **PhoneKeyPad**, which prompts user for a String (case insensitive), and converts to a sequence of keypad digits. Use (a) a nested-if, (b) a switch-case-default.

Hints

1. You can use in.next().toLowerCase() to read a String and convert it to lowercase to reduce your cases.
2. In switch-case, you can handle multiple cases by omitting the break statement, e.g.,

```
switch (inChar) {
    case 'a': case 'b': case 'c': // No break for 'a' and 'b', fall thru 'c'
        System.out.print(2); break;
    case 'd': case 'e': case 'f':
        .....
    default:
        .....
}
```

7.4 Caesar's Code (String & char)

Caesar's Code is one of the simplest encryption techniques. Each letter in the plaintext is replaced by a letter some fixed number of position (n) down the alphabet cyclically. In this exercise, we shall pick n=3. That is, 'A' is replaced by 'D', 'B' by 'E', 'C' by 'F', ..., 'X' by 'A', ..., 'Z' by 'C'.

Write a program called **CaesarCode** to cipher the Caesar's code. The program shall prompt user for a plaintext string consisting of mix-case letters only; compute the ciphertext; and print the ciphertext in uppercase. For example,

Enter a plaintext string: **Testing**
The ciphertext string is: WHVWLQJ

Hints

1. Use `in.next().toUpperCase()` to read an input string and convert it into uppercase to reduce the number of cases.
2. You can use a big nested-if with 26 cases ('A'-'Z'). But it is much better to consider 'A' to 'W' as one case; 'X', 'Y' and 'Z' as 3 separate cases.
3. Take note that char 'A' is represented as Unicode number 65 and char 'D' as 68. However, 'A' + 3 gives 68. This is because `char + int` is implicitly casted to `int + int` which returns an `int` value. To obtain a char value, you need to perform explicit type casting using `(char)('A' + 3)`. Try printing ('A' + 3) with and without type casting.

7.5 Decipher Caesar's Code (String & char)

Write a program called **DecipherCaesarCode** to decipher the Caesar's code described in the previous exercise. The program shall prompts user for a ciphertext string consisting of mix-case letters only; compute the plaintext; and print the plaintext in uppercase. For example,

Enter a ciphertext string: **wHVwLQJ**
The plaintext string is: TESTING

7.6 Exchange Cipher (String & char)

This simple cipher exchanges 'A' and 'Z', 'B' and 'Y', 'C' and 'X', and so on.

Write a program called **ExchangeCipher** that prompts user for a plaintext string consisting of mix-case letters only. Your program shall compute the ciphertext; and print the ciphertext in uppercase. For examples,

Enter a plaintext string: **abcXYZ**
The ciphertext string is: ZYXCBA

Hints

1. Use `in.next().toUpperCase()` to read an input string and convert it into uppercase to reduce the number of cases.
2. You can use a big nested-if with 26 cases ('A'-'Z'), or use the following relationship:

`'A' + 'Z' == 'B' + 'Y' == 'C' + 'X' == ... == plainTextChar + cipherTextChar`
Hence, `cipherTextChar = 'A' + 'Z' - plainTextChar`

7.7 TestPalindromicWord and TestPalindromicPhrase (String & char)

A word that reads the same backward as forward is called a palindrome, e.g., "mom", "dad", "racecar", "madam", and "Radar" (case-insensitive).

Write a program called **TestPalindromicWord**, that prompts user for a word and prints "'xxx' is|is not a palindrome".

A phrase that reads the same backward as forward is also called a palindrome, e.g., "Madam, I'm Adam", "A man, a plan, a canal - Panama!" (ignoring punctuation and capitalization).

Modify your program (called **TestPalindromicPhrase**) to check for palindromic phrase. Use `in.nextLine()` to read a line of input.

Hints

1. **Maintain two indexes, forwardIndex (fIdx) and backwardIndex (bIdx), to scan the phrase forward and backward.**

```
int fIdx = 0, bIdx = strLen - 1;
```

```

while (fIdx < bIdx) {
    .....
    ++fIdx;
    --bIdx;
}
// or
for (int fIdx = 0, bIdx = strLen - 1; fIdx < bIdx; ++fIdx, --bIdx) {
    .....
}

```

2. You can check if a char *c* is a letter either using built-in boolean function `Character.isLetter(c)`; or boolean expression `(c >= 'a' && c <= 'z')`. Skip the index if it does not contain a letter.

7.8 CheckBinStr (String & char)

The binary number system uses 2 symbols, 0 and 1. Write a program called **CheckBinStr** to verify a binary string. The program shall prompt user for a binary string; and decide if the input string is a valid binary string. For example,

```

Enter a binary string: 10101100
"10101100" is a binary string

```

```

Enter a binary string: 10120000
"10120000" is NOT a binary string

```

Hints

Use the following coding pattern which involves a boolean flag to check the input string.

```

// Declare variables
String inStr;    // The input string
int inStrLen;    // The length of the input string
char inChar;     // Each char of the input string
boolean isValid; // "is" or "is not" a valid binary string?
.....

isValid = true; // Assume that the input is valid, unless our check fails
for (.....) {
    inChar = .....;
    if (!(inChar == '0' || inChar == '1')) {
        isValid = false;
        break; // break the loop upon first error, no need to continue for more errors
               // If this is not encountered, isValid remains true after the loop.
    }
}
if (isValid) {
    System.out.println(.....);
} else {
    System.out.println(.....);
}
// or using one liner
//System.out.println(isValid ? ... : ...);

```

7.9 CheckHexStr (String & char)

The hexadecimal (hex) number system uses 16 symbols, 0-9 and A-F (or a-f). Write a program to verify a hex string. The program shall prompt user for a hex string; and decide if the input string is a valid hex string. For examples,

```

Enter a hex string: 123aBc
"123aBc" is a hex string

```

```

Enter a hex string: 123aBcx

```

"123aBcx" is NOT a hex string

Hints

```
if (!(inChar >= '0' && inChar <= '9')
    || (inChar >= 'A' && inChar <= 'F')
    || (inChar >= 'a' && inChar <= 'f')) { // Use positive logic and then reverse
    .....
}
```

7.10 Bin2Dec (String & char)

Write a program called **Bin2Dec** to convert an input binary string into its equivalent decimal number. Your output shall look like:

Enter a Binary string: **1011**
The equivalent decimal number for binary "1011" is: 11

Enter a Binary string: **1234**
error: invalid binary string "1234"

Hints

See "Code Example".

7.11 Hex2Dec (String & char)

Write a program called **Hex2Dec** to convert an input hexadecimal string into its equivalent decimal number. Your output shall look like:

Enter a Hexadecimal string: **1a**
The equivalent decimal number for hexadecimal "1a" is: 26

Enter a Hexadecimal string: **1y3**
error: invalid hexadecimal string "1y3"

Hints

See "Code Example".

7.12 Oct2Dec (String & char)

Write a program called **Oct2Dec** to convert an input Octal string into its equivalent decimal number. For example,

Enter an Octal string: **147**
The equivalent decimal number "147" is: 103

8. Exercises on Arrays

8.1 PrintArray (Array)

Write a program called **PrintArray** which prompts user for the number of items in an array (a non-negative integer), and saves it in an int variable called **NUM_ITEMS**. It then prompts user for the values of all the items and saves them in an int array called **items**. The program shall then print the contents of the array in the form of [x1, x2, ..., xn]. For example,

Enter the number of items: **5**
Enter the value of all items (separated by space): **3 2 5 6 9**
The values are: [3, 2, 5, 6, 9]

Hints

```

// Declare variables
final int NUM_ITEMS;
int[] items; // Declare array name, to be allocated after NUM_ITEMS is known
.....

// Prompt for the number of items and read the input as "int"
.....
NUM_ITEMS = .....

// Allocate the array
items = new int[NUM_ITEMS];

// Prompt and read the items into the "int" array, if array length > 0
if (items.length > 0) {
    .....
    for (int i = 0; i < items.length; ++i) { // Read all items
        .....
    }
}

// Print array contents, need to handle first item and subsequent items differently
.....
for (int i = 0; i < items.length; ++i) {
    if (i == 0) {
        // Print the first item without a leading commas
        .....
    } else {
        // Print the subsequent items with a leading commas
        .....
    }
    // or, using a one liner
    //System.out.print((i == 0) ? ..... : .....);
}

```

8.2 PrintArrayInStars (Array)

Write a program called **printArrayInStars** which prompts user for the number of items in an array (a non-negative integer), and saves it in an int variable called NUM_ITEMS. It then prompts user for the values of all the items (non-negative integers) and saves them in an int array called items. The program shall then print the contents of the array in a graphical form, with the array index and values represented by number of stars. For examples,

```

Enter the number of items: 5
Enter the value of all items (separated by space): 7 4 3 0 7
0: ***** (7)
1: **** (4)
2: *** (3)
3: (0)
4: ***** (7)

```

Hints

```

// Declare variables
final int NUM_ITEMS;
int[] items; // Declare array name, to be allocated after NUM_ITEMS is known
.....
.....
// Print array in "index: number of stars" using a nested-loop
// Take note that rows are the array indexes and columns are the value in that index
for (int idx = 0; idx < items.length; ++idx) { // row
    System.out.print(idx + ": ");
    // Print value as the number of stars

```

```

    for (int starNo = 1; starNo <= items[idx]; ++starNo) { // column
        System.out.print("*");
    }
    .....
}
.....

```

8.3 GradesStatistics (Array)

Write a program which prompts user for the number of students in a class (a non-negative integer), and saves it in an int variable called numStudents. It then prompts user for the grade of each of the students (integer between 0 to 100) and saves them in an int array called grades. The program shall then compute and print the average (in double rounded to 2 decimal places) and minimum/maximum (in int).

```

Enter the number of students: 5
Enter the grade for student 1: 98
Enter the grade for student 2: 78
Enter the grade for student 3: 78
Enter the grade for student 4: 87
Enter the grade for student 5: 76
The average is: 83.40
The minimum is: 76
The maximum is: 98

```

8.4 Hex2Bin (Array for Table Lookup)

Write a program called **Hex2Bin** that prompts user for a hexadecimal string and print its equivalent binary string. The output shall look like:

```

Enter a Hexadecimal string: 1abc
The equivalent binary for hexadecimal "1abc" is: 0001 1010 1011 1100

```

Hints

1. Use an array of 16 Strings containing binary strings corresponding to hexadecimal number 0-9A-F (or a-f), as follows

```

final String[] HEX_BITS = {"0000", "0001", "0010", "0011",
    "0100", "0101", "0110", "0111",
    "1000", "1001", "1010", "1011",
    "1100", "1101", "1110", "1111"};

```

8.5 Dec2Hex (Array for Table Lookup)

Write a program called **Dec2Hex** that prompts user for a positive decimal number, read as int, and print its equivalent hexadecimal string. The output shall look like:

```

Enter a decimal number: 1234
The equivalent hexadecimal number is 4D2

```

9. Exercises on Methods

9.1 exponent() (method)

Write a method called exponent(int base, int exp) that returns an int value of base raises to the power of exp. The signature of the method is:

```

public static int exponent(int base, int exp);

```

Assume that exp is a non-negative integer and base is an integer. Do not use any Math library functions. Also write the main() method that prompts user for the base and exp; and prints the result. For example,

Enter the base: 3
Enter the exponent: 4
3 raises to the power of 4 is: 81

Hints

```
.....  
public class Exponent {  
    public static void main(String[] args) {  
        // Declare variables  
        int exp; // exponent (non-negative integer)  
        int base; // base (integer)  
        .....  
        // Prompt and read exponent and base  
        .....  
        // Print result  
        System.out.println(base + " raises to the power of " + exp + " is: " + exponent(base, exp));  
    }  
  
    // Returns "base" raised to the power "exp"  
    public static int exponent(int base, int exp) {  
        int product = 1; // resulting product  
  
        // Multiply product and base for exp number of times  
        for (.....) {  
            product *= base;  
        }  
  
        return product;  
    }  
}
```

9.2 isOdd() (method)

Write a boolean method called isOdd() in a class called **OddEvenTest**, which takes an int as input and returns true if it is odd. The signature of the method is as follows:

```
public static boolean isOdd(int number);
```

Also write the main() method that prompts user for a number, and prints "ODD" or "EVEN". You should test for negative input. For examples,

Enter a number: 9
9 is an odd number

Enter a number: 8
8 is an even number

Enter a number: -5
-5 is an odd number

Hints

See Notes.

9.3 hasEight() (method)

Write a boolean method called hasEight(), which takes an int as input and returns true if the number contains the digit 8 (e.g., 18, 168, 1288). The signature of the method is as follows:

```
public static boolean hasEight(int number);
```

Write a program called **MagicSum**, which prompts user for integers (or -1 to end), and produce the sum of numbers containing the digit 8. Your program should use the above methods. A sample output of the program is

as follows:

```
Enter a positive integer (or -1 to end): 1
Enter a positive integer (or -1 to end): 2
Enter a positive integer (or -1 to end): 3
Enter a positive integer (or -1 to end): 8
Enter a positive integer (or -1 to end): 88
Enter a positive integer (or -1 to end): -1
The magic sum is: 96
```

Hints

The coding pattern to repeat until input is -1 (called sentinel value) is:

```
final int SENTINEL = -1; // Terminating input
int number;

// Read first input to "seed" the while loop
System.out.print("Enter a positive integer (or -1 to end): ");
number = in.nextInt();

while (number != SENTINEL) { // Repeat until input is -1
    .....
    .....

    // Read next input. Repeat if the input is not the SENTINEL
    // Take note that you need to repeat these codes!
    System.out.print("Enter a positive integer (or -1 to end): ");
    number = in.nextInt();
}
```

You can either repeatably use modulus/divide ($n\%10$ and $n=n/10$) to extract and drop each digit in int; or convert the int to String and use the String's `charAt()` to inspect each char.

9.4 `print()` (Array & Method)

Write a method called **`print()`**, which takes an int array and print its contents in the form of [a1, a2, ..., an]. Take note that there is no comma after the last element. The method's signature is as follows:

```
public static void print(int[] array);
```

Also write a test driver to test this method (you should test on empty array, one-element array, and n-element array).

How to handle `double[]` or `float[]`? You need to write a overloaded version for `double[]` and a overloaded version for `float[]`, with the following signatures:

```
public static void print(double[] array)
public static void print(float[] array)
```

The above is known as method overloading, where the same method name can have many versions, differentiated by its parameter list.

Hints

For the first element, print its value; for subsequent elements, print commas followed by the value.

9.5 `arrayToString()` (Array & Method)

Write a method called **`arrayToString()`**, which takes an int array and return a String in the form of [a1, a2, ..., an]. Take note that this method returns a String, the previous exercise returns void but prints the output. The method's signature is as follows:

```
public static String arrayToString(int[] array);
```

Also write a test driver to test this method (you should test on empty array, one-element array, and n-element array).

Notes: This is similar to the built-in function `Arrays.toString()`. You could study its source code.

9.6 contains() (Array & Method)

Write a boolean method called **contains()**, which takes an array of int and an int; and returns true if the array contains the given int. The method's signature is as follows:

```
public static boolean contains(int[] array, int key);
```

Also write a test driver to test this method.

9.7 search() (Array & Method)

Write a method called **search()**, which takes an array of int and an int; and returns the array index if the array contains the given int; or -1 otherwise. The method's signature is as follows:

```
public static int search(int[] array, int key);
```

Also write a test driver to test this method.

9.8 equals() (Array & Method)

Write a boolean method called **equals()**, which takes two arrays of int and returns true if the two arrays are exactly the same (i.e., same length and same contents). The method's signature is as follows:

```
public static boolean equals(int[] array1, int[] array2)
```

Also write a test driver to test this method.

9.9 copyOf() (Array & Method)

Write a boolean method called **copyOf()**, which takes an int Array and returns a copy of the given array. The method's signature is as follows:

```
public static int[] copyOf(int[] array)
```

Also write a test driver to test this method.

Write another version for **copyOf()** which takes a second parameter to specify the length of the new array. You should truncate or pad with zero so that the new array has the required length.

```
public static int[] copyOf(int[] array, int newLength)
```

NOTES: This is similar to the built-in function `Arrays.copyOf()`.

9.10 swap() (Array & Method)

Write a method called **swap()**, which takes two arrays of int and swap their contents if they have the same length. It shall return true if the contents are successfully swapped. The method's signature is as follows:

```
public static boolean swap(int[] array1, int[] array2)
```

Also write a test driver to test this method.

Hints

You need to use a temporary location to swap two storage locations.

```
// Swap item1 and item2
int item1, item2, temp;
temp = item1;
item1 = item2;
item2 = item1;
```

```
// You CANNOT simply do: item1 = item2; item2 = item2;
```

9.11 reverse() (Array & Method)

Write a method called **reverse()**, which takes an array of int and reverse its contents. For example, the reverse of [1,2,3,4] is [4,3,2,1]. The method's signature is as follows:

```
public static void reverse(int[] array)
```

Take note that the array passed into the method can be modified by the method (this is called "pass by reference"). On the other hand, primitives passed into a method cannot be modified. This is because a clone is created and passed into the method instead of the original copy (this is called "pass by value").

Also write a test driver to test this method.

Hints

You might use two indexes in the loop, one moving forward and one moving backward to point to the two elements to be swapped.

```
for (int fIdx = 0, bIdx = array.length - 1; fIdx < bIdx; ++fIdx, --bIdx) {  
    // Swap array[fIdx] and array[bIdx]  
    // Only need to transverse half of the array elements  
}
```

You need to use a temporary location to swap two storage locations.

```
// Swap item1 and item2  
int item1, item2, temp;  
temp = item1;  
item1 = item2;  
item2 = temp;  
// You CANNOT simply do: item1 = item2; item2 = item1;
```

9.12 GradesStatistics (Array & Method)

Write a program called **GradesStatistics**, which reads in n grades (of int between 0 and 100, inclusive) and displays the average, minimum, maximum, median and standard deviation. Display the floating-point values upto 2 decimal places. Your output shall look like:

```
Enter the number of students: 4  
Enter the grade for student 1: 50  
Enter the grade for student 2: 51  
Enter the grade for student 3: 56  
Enter the grade for student 4: 53  
The grades are: [50, 51, 56, 53]  
The average is: 52.50  
The median is: 52.00  
The minimum is: 50  
The maximum is: 56  
The standard deviation is: 2.29
```

The formula for calculating standard deviation is:

$$\sigma = \sqrt{\frac{1}{n} \sum_{i=0}^{n-1} x_i^2 - \mu^2}, \text{ where } \mu \text{ is the mean}$$

Hints:

```
public class GradesStatistics {  
    public static int[] grades; // Declare an int[], to be allocated later.  
    // This array is accessible by all the methods.  
  
    public static void main(String[] args) {  
        readGrades(); // Read and save the inputs in global int[] grades  
    }  
}
```

```

        System.out.println("The grades are: ");
        print(grades);
        System.out.println("The average is " + average(grades));
        System.out.println("The median is " + median(grades));
        System.out.println("The minimum is " + min(grades));
        System.out.println("The maximum is " + max(grades));
        System.out.println("The standard deviation is " + stdDev(grades));
    }

    // Prompt user for the number of students and allocate the global "grades" array.
    // Then, prompt user for grade, check for valid grade, and store in "grades".
    public static void readGrades() { ..... }

    // Print the given int array in the form of [x1, x2, x3,..., xn].
    public static void print(int[] array) { ..... }

    // Return the average value of the given int[]
    public static double average(int[] array) { ..... }

    // Return the median value of the given int[]
    // Median is the center element for odd-number array,
    // or average of the two center elements for even-number array.
    // Use Arrays.sort(anArray) to sort anArray in place.
    public static double median(int[] array) { ..... }

    // Return the maximum value of the given int[]
    public static int max(int[] array) {
        int max = array[0]; // Assume that max is the first element
        // From second element, if the element is more than max, set the max to this element.
        .....
    }

    // Return the minimum value of the given int[]
    public static int min(int[] array) { ..... }

    // Return the standard deviation of the given int[]
    public static double stdDev(int[] array) { ..... }
}

```

Take note that besides readGrade() that relies on global variable grades, all the methods are self-contained general utilities that operate on any given array.

9.13 GradesHistogram (Array & Method)

Write a program called **GradesHistogram**, which reads in n grades (as in the previous exercise), and displays the horizontal and vertical histograms. For example:

```

0 - 9: ***
10 - 19: ***
20 - 29:
30 - 39:
40 - 49: *
50 - 59: *****
60 - 69:
70 - 79:
80 - 89: *
90 - 100: **

```

```

*
*

```

```

*   *           *
*   *           *           *
*   *           *   *           *   *

```

0-9 10-19 20-29 30-39 40-49 50-59 60-69 70-79 80-89 90-100

10. Exercises on Command-line Arguments, Recursion

10.1 Arithmetic (Command-Line Arguments)

Write a program called **Arithmetic** that takes three command-line arguments: two integers followed by an arithmetic operator (+, -, * or /). The program shall perform the corresponding operation on the two integers and print the result. For example:

```

java Arithmetic 3 2 +
3+2=5

```

```

java Arithmetic 3 2 -
3-2=1

```

```

java Arithmetic 3 2 /
3/2=1

```

Hints

The method `main(String[] args)` takes an argument: "an array of String", which is often (but not necessary) named `args`. This parameter captures the command-line arguments supplied by the user when the program is invoked. For example, if a user invokes:

```

java Arithmetic 12345 4567 +

```

The three command-line arguments "12345", "4567" and "+" will be captured in a String array {"12345", "4567", "+"} and passed into the `main()` method as the argument `args`. That is,

```

args is: {"12345", "4567", "+"} // args is a String array
args.length is: 3 // length of the array
args[0] is: "12345" // 1st element of the String array
args[1] is: "4567" // 2nd element of the String array
args[2] is: "+" // 3rd element of the String array
args[0].length() is: 5 // length of 1st String element
args[1].length() is: 4 // length of the 2nd String element
args[2].length() is: 1 // length of the 3rd String element

```

```

public class Arithmetic {
    public static void main (String[] args) {
        int operand1, operand2;
        char theOperator;

        // Check if there are 3 command-line arguments in the
        // String[] args by using length variable of array.
        if (args.length != 3) {
            System.err.println("Usage: java Arithmetic int1 int2 op");
            return;
        }

        // Convert the 3 Strings args[0], args[1], args[2] to int and char.
        // Use the Integer.parseInt(aStr) to convert a String to an int.
        operand1 = Integer.parseInt(args[0]);
        operand2 = .....

        // Get the operator, assumed to be the first character of
        // the 3rd string. Use method charAt() of String.
        theOperator = args[2].charAt(0);
        System.out.print(args[0] + args[2] + args[1] + "=");
    }
}

```

```

switch(theOperator) {
    case ('-'): System.out.println(operand1 - operand2); break;
    case ('+'): .....
    case ('*'): .....
    case ('/'): .....
    default:
        System.err.println("Error: invalid operator!");
}
}
}

```

Notes:

- To provide command-line arguments, use the "cmd" or "terminal" to run your program in the form "java ClassName arg1 arg2".
- To provide command-line arguments in Eclipse, right click the source code \Rightarrow "Run As" \Rightarrow "Run Configurations..." \Rightarrow Select "Main" and choose the proper main class \Rightarrow Select "Arguments" \Rightarrow Enter the command-line arguments, e.g., "3 2 +" in "Program Arguments".
- To provide command-line arguments in NetBeans, right click the "Project" name \Rightarrow "Set Configuration" \Rightarrow "Customize..." \Rightarrow Select categories "Run" \Rightarrow Enter the command-line arguments, e.g., "3 2 +" in the "Arguments" box (but make sure you select the proper Main class).

Question: Try "java Arithmetic 2 4 *" (in CMD shell and Eclipse/NetBeans) and explain the result obtained. How to resolve this problem?

In Windows' CMD shell, * is known as a wildcard character, that expands to give the list of file in the directory (called Shell Expansion). For example, "dir *.java" lists all the file with extension of ".java". You could double-quote the * to prevent shell expansion. Eclipse has a bug in handling this, even * is double-quoted. NetBeans??

SumDigits (Command-line Arguments)

Write a program called **SumDigits** to sum up the individual digits of a positive integer, given in the command line. The output shall look like:

```
java SumDigits 12345
```

```
The sum of digits = 1 + 2 + 3 + 4 + 5 = 15
```

Exercises on Recursion

In programming, a recursive function (or method) calls itself. The classical example is factorial(n), which can be defined recursively as $f(n)=n*f(n-1)$. Nonetheless, it is important to take note that a recursive function should have a terminating condition (or base case), in the case of factorial, $f(0)=1$. Hence, the full definition is:

factorial(n) = 1, for n = 0

factorial(n) = n * factorial(n-1), for all n > 1

For example, suppose n = 5:

```

// Recursive call
factorial(5) = 5 * factorial(4)
factorial(4) = 4 * factorial(3)
factorial(3) = 3 * factorial(2)
factorial(2) = 2 * factorial(1)
factorial(1) = 1 * factorial(0)
factorial(0) = 1 // Base case
// Unwinding
factorial(1) = 1 * 1 = 1
factorial(2) = 2 * 1 = 2
factorial(3) = 3 * 2 = 6
factorial(4) = 4 * 6 = 24
factorial(5) = 5 * 24 = 120 (DONE)

```

10.2 Factorial Recursive

Write a recursive method called factorial() to compute the factorial of the given integer.

```
public static int factorial(int n)
```

The recursive algorithm is:

factorial(n) = 1, if n = 0

factorial(n) = n * factorial(n-1), if n > 0

Compare your code with the iterative version of the factorial():

factorial(n) = 1*2*3*...*n

Hints

Writing recursive function is straight forward. You simply translate the recursive definition into code with return.

// Return the factorial of the given integer, recursively

```
public static int factorial(int n) {  
    if (n == 0) {  
        return 1; // base case  
    } else {  
        return n * factorial(n-1); // call itself  
    }  
    // or one liner  
    // return (n == 0) ? 1 : n*factorial(n-1);  
}
```

Notes

1. Recursive version is often much shorter.
2. The recursive version uses much more computational and storage resources, and it need to save its current states before each successive recursive call, so as to unwind later.

10.3 Fibonacci (Recursive)

Write a recursive method to compute the Fibonacci number of n, defined as follows:

F(0) = 0

F(1) = 1

F(n) = F(n-1) + F(n-2) for n >= 2

Compare the recursive version with the iterative version written earlier.

Hints

// Translate the recursive definition into code with return statements

```
public static int fibonacci(int n) {  
    if (n == 0) {  
        return 0;  
    } else if (n == 1) {  
        return 1;  
    } else {  
        return fibonacci(n-1) + fibonacci(n-2);  
    }  
}
```

10.4 Length of a Running Number Sequence (Recursive)

A special number sequence is defined as follows:

S(1) = 1

S(2) = 12

```

S(3) = 123
S(4) = 1234
.....
S(9) = 123456789    // length is 9
S(10) = 12345678910  // length is 11
S(11) = 1234567891011 // length is 13
S(12) = 123456789101112 // length is 15
.....

```

Write a recursive method to compute the length of S(n), defined as follows:

```

len(1) = 1
len(n) = len(n-1) + numOfDigits(n)

```

Also write an iterative version.

10.5 GCD (Recursive)

Write a recursive method called gcd() to compute the greatest common divisor of two given integers.

```

public static void gcd(int a, int b)

gcd(a,b) = a, if b = 0
gcd(a,b) = gcd(b, remainder(a,b)), if b > 0

```

11. More (Difficult) Exercises

11.1 JDK Source Code

Extract the source code of the class Math from the JDK source code (JDK Installed Directory ⇒ "lib" ⇒ "src.zip" ⇒ "java.base" ⇒ "java" ⇒ "lang" ⇒ "Math.java"). Study how constants such as E and PI are defined. Also study how methods such as abs(), max(), min(), toDegree(), etc, are written.

Also study the "Integer.java", "String.java".

11.2 Matrices (2D Arrays)

Similar to Math class, write a Matrix library that supports matrix operations (such as addition, subtraction, multiplication) via 2D arrays. The operations shall support both double and int. Also write a test class to exercise all the operations programmed.

Hints

```

public class Matrix {
    // Method signatures
    public static void print(int[][] m);
    public static void print(double[][] m);
    public static boolean haveSameDimension(int[][] m1, int[][] m2); // Used in add(), subtract()
    public static boolean haveSameDimension(double[][] m1, double[][] m2);
    public static int[][] add(int[][] m1, int[][] m2);
    public static double[][] add(double[][] m1, double[][] m2);
    public static int[][] subtract(int[][] m1, int[][] m2);
    public static double[][] subtract(double[][] m1, double[][] m2);
    public static int[][] multiply(int[][] m1, int[][] m2);
    public static double[][] multiply(double[][] m1, double[][] m2);
    .....
}

```

11.3 PrintAnimalPattern (Special Characters and Escape Sequences)

Write a program called **PrintAnimalPattern**, which uses println() to produce this pattern:

```

' _ '

```



```

    (©©)
  /=====V
 / || %% ||
* ||---||
¥¥  ¥¥
""  ""

```

Hints

Use escape sequence \uhhhh where hhhh are four hex digits to display Unicode characters such as ¥ and ©. ¥ is 165 (00A5H) and © is 169 (00A9H) in both ISO-8859-1 (Latin-1) and Unicode character sets.

Double-quote (") and black-slash (\) require escape sequence inside a String. Single quote (') does not require escape sign.

Try

Print the same pattern using printf(). (Hints: Need to use %% to print a % in printf() because % is the suffix for format specifier.)

11.4 Print Patterns (nested-loop)

Write a method to print each of the followings patterns using nested loops in a class called **PrintPatterns**. The program shall prompt user for the size of the pattern. The signatures of the methods are:

```
public static void printPatternX(int size); // X: A, B, C,...; size is a positive integer.
```

```

#####      #      #
#####      ###      ###
#####      #####      #####
#####      #####      #####
###      #####      #####
#      #####      #####
(a)      (b)      #####

```

```

#####
#####
###
#
(c)

```

```

1      1 2 3 4 5 6 7 8      1 8 7 6 5 4 3 2 1
1 2      1 2 3 4 5 6 7      2 1 7 6 5 4 3 2 1
1 2 3      1 2 3 4 5 6      3 2 1 6 5 4 3 2 1
1 2 3 4      1 2 3 4 5      4 3 2 1 5 4 3 2 1
1 2 3 4 5      1 2 3 4      5 4 3 2 1 4 3 2 1
1 2 3 4 5 6      1 2 3      6 5 4 3 2 1 3 2 1
1 2 3 4 5 6 7      1 2      7 6 5 4 3 2 1 2 1
1 2 3 4 5 6 7 8      1      8 7 6 5 4 3 2 1 1
(d)      (e)      (f)      (g)

```

```

      1      1 2 3 4 5 6 7 8 7 6 5 4 3 2 1
1 2 1      1 2 3 4 5 6 7 6 5 4 3 2 1

```

```

    1 2 3 2 1      1 2 3 4 5 6 5 4 3 2 1
    1 2 3 4 3 2 1    1 2 3 4 5 4 3 2 1
    1 2 3 4 5 4 3 2 1    1 2 3 4 3 2 1
    1 2 3 4 5 6 5 4 3 2 1    1 2 3 2 1
    1 2 3 4 5 6 7 6 5 4 3 2 1    1 2 1
    1 2 3 4 5 6 7 8 7 6 5 4 3 2 1    1
    (h)                      (i)

```

```

1          1 1 2 3 4 5 6 7 8 7 6 5 4 3 2 1
1 2        2 1 1 2 3 4 5 6 7 7 6 5 4 3 2 1
1 2 3      3 2 1 1 2 3 4 5 6 6 5 4 3 2 1
1 2 3 4    4 3 2 1 1 2 3 4 5 5 4 3 2 1
1 2 3 4 5  5 4 3 2 1 1 2 3 4 4 3 2 1
1 2 3 4 5 6 6 5 4 3 2 1 1 2 3 3 2 1
1 2 3 4 5 6 7 7 6 5 4 3 2 1 1 2 2 1
1 2 3 4 5 6 7 8 7 6 5 4 3 2 1 1 1
    (j)                      (k)

```

```

    1
   2 3 2
  3 4 5 4 3
 4 5 6 7 6 5 4
5 6 7 8 9 8 7 6 5
6 7 8 9 0 1 0 9 8 7 6
7 8 9 0 1 2 3 2 1 0 9 8 7
8 9 0 1 2 3 4 5 4 3 2 1 0 9 8
    (l)

```

11.5 Print Triangles (nested-loop)

Write a method to print each of the following patterns using nested-loops in a class called **PrintTriangles**. The program shall prompt user for the number of rows. The signatures of the methods are:

```
public static void printXxx(int numRows); // Xxx is the pattern's name
```

```

    1
   1 2 1
  1 2 4 2 1
 1 2 4 8 4 2 1
1 2 4 8 16 8 4 2 1
1 2 4 8 16 32 16 8 4 2 1
1 2 4 8 16 32 64 32 16 8 4 2 1
1 2 4 8 16 32 64 128 64 32 16 8 4 2 1
    (a) PowerOf2Triangle

```

```

1          1
1 1        1 1
1 2 1      1 2 1
1 3 3 1    1 3 3 1
1 4 6 4 1  1 4 6 4 1

```

```

1 5 10 10 5 1      1 5 10 10 5 1
1 6 15 20 15 6 1    1 6 15 20 15 6 1
(b) PascalTriangle1  (c) PascalTriangle2

```

11.6 Trigonometric Series

Write a method to compute $\sin(x)$ and $\cos(x)$ using the following series expansion, in a class called **TrigonometricSeries**. The signatures of the methods are:

```

public static double sin(double x, int numTerms); // x in radians, NOT degrees
public static double cos(double x, int numTerms);

```

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \dots$$

$$\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!} - \dots$$

Compare the values computed using the series with the JDK methods `Math.sin()`, `Math.cos()` at $x=0$, $\pi/6$, $\pi/4$, $\pi/3$, $\pi/2$ using various numbers of terms.

Hints

Do not use `int` to compute the factorial; as factorial of 13 is outside the `int` range. Avoid generating large numerator and denominator. Use `double` to compute the terms as:

$$\frac{x^n}{n!} = \left(\frac{x}{n}\right) \left(\frac{x}{n-1}\right) \dots \left(\frac{x}{1}\right)$$

11.7 Exponential Series

Write a method to compute e and $\exp(x)$ using the following series expansion, in a class called **ExponentialSeries**. The signatures of the methods are:

```

public static double exp(int numTerms); // x in radians
public static double exp(double x, int numTerms);

```

$$e = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!} + \dots$$

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \dots$$

11.8 Special Series

Write a method to compute the sum of the series in a class called **SpecialSeries**. The signature of the method is:

```

public static double specialSeries(double x, int numTerms);

```

$$x + \frac{1}{2} \times \frac{x^3}{3} + \frac{1 \times 3}{2 \times 4} \times \frac{x^5}{5} + \frac{1 \times 3 \times 5}{2 \times 4 \times 6} \times \frac{x^7}{7} + \frac{1 \times 3 \times 5 \times 7}{2 \times 4 \times 6 \times 8} \times \frac{x^9}{9} + \dots; -1 \leq x \leq 1$$

11.9 FactorialInt (Handling Overflow)

Write a program called **FactorialInt** to list all the factorials that can be expressed as an `int` (i.e., 32-bit signed integer in the range of $[-2147483648, 2147483647]$). Your output shall look like:

```

The factorial of 1 is 1
The factorial of 2 is 2
...
The factorial of 12 is 479001600
The factorial of 13 is out of range

```

Hints

The maximum and minimum values of a 32-bit int are kept in constants `Integer.MAX_VALUE` and `Integer.MIN_VALUE`, respectively. Try these statements:

```
System.out.println(Integer.MAX_VALUE);
System.out.println(Integer.MIN_VALUE);
System.out.println(Integer.MAX_VALUE + 1);
```

Take note that in the third statement, Java Runtime does not flag out an overflow error, but silently wraps the number around. Hence, you cannot use $F(n) * (n+1) > \text{Integer.MAX_VALUE}$ to check for overflow. Instead, overflow occurs for $F(n+1)$ if $(\text{Integer.MAX_VALUE} / \text{Factorial}(n)) < (n+1)$, i.e., no more room for the next number.

Try

Modify your program called **FactorialLong** to list all the factorial that can be expressed as a long (64-bit signed integer). The maximum value for long is kept in a constant called `Long.MAX_VALUE`.

11.10 FibonacciInt (Handling Overflow)

Write a program called **FibonacciInt** to list all the Fibonacci numbers, which can be expressed as an int (i.e., 32-bit signed integer in the range of [-2147483648, 2147483647]). The output shall look like:

```
F(0) = 1
F(1) = 1
F(2) = 2
...
F(45) = 1836311903
F(46) is out of the range of int
```

Hints

The maximum and minimum values of a 32-bit int are kept in constants `Integer.MAX_VALUE` and `Integer.MIN_VALUE`, respectively. Try these statements:

```
System.out.println(Integer.MAX_VALUE);
System.out.println(Integer.MIN_VALUE);
System.out.println(Integer.MAX_VALUE + 1);
```

Take note that in the third statement, Java Runtime does not flag out an overflow error, but silently wraps the number around. Hence, you cannot use $F(n) = F(n-1) + F(n-2) > \text{Integer.MAX_VALUE}$ to check for overflow. Instead, overflow occurs for $F(n)$ if $\text{Integer.MAX_VALUE} - F(n-1) < F(n-2)$ (i.e., no more room for the next Fibonacci number).

Try

Write a similar program called **TribonacciInt** for Tribonacci numbers.

11.11 Number System Conversion

Write a method call **toRadix()** which converts a positive integer from one radix into another. The method has the following header:

```
public static String toRadix(String in, int inRadix, int outRadix) // The input and output are treated as String.
```

Write a program called **NumberConversion**, which prompts the user for an input string, an input radix, and an output radix, and display the converted number. The output shall look like:

```
Enter a number and radix: A1B2
Enter the input radix: 16
Enter the output radix: 2
```

"A1B2" in radix 16 is "1010000110110010" in radix 2.

11.12 NumberGuess

Write a program called **NumberGuess** to play the number guessing game. The program shall generate a random number between 0 and 99. The player inputs his/her guess, and the program shall response with "Try higher", "Try lower" or "You got it in n trials" accordingly. For example:

```
java NumberGuess
```

```
Key in your guess:
```

```
50
```

```
Try higher
```

```
70
```

```
Try lower
```

```
65
```

```
Try lower
```

```
61
```

```
You got it in 4 trials!
```

Hints

Use `Math.random()` to produce a random number in double between 0.0 (inclusive) and 1.0 (exclusive). To produce an int between 0 and 99, use:

```
final int SECRET_NUMBER = (int)(Math.random()*100); // truncate to int
```

11.13 WordGuess

Write a program called **WordGuess** to guess a word by trying to guess the individual characters. The word to be guessed shall be provided using the command-line argument. Your program shall look like:

```
java WordGuess testing
```

```
Key in one character or your guess word: t
```

```
Trial 1: t__t__
```

```
Key in one character or your guess word: g
```

```
Trial 2: t__t__g
```

```
Key in one character or your guess word: e
```

```
Trial 3: te_t__g
```

```
Key in one character or your guess word: testing
```

```
Congratulation!
```

```
You got in 4 trials
```

Hints

Set up a boolean array (of the length of the word to be guessed) to indicate the positions of the word that have been guessed correctly.

Check the length of the input String to determine whether the player enters a single character or a guessed word. If the player enters a single character, check it against the word to be guessed, and update the boolean array that keeping the result so far.

Try

Try retrieving the word to be guessed from a text file (or a dictionary) randomly.

11.14 DateUtil

Complete the following methods in a class called **DateUtil**:

- `boolean isLeapYear(int year)`: returns true if the given year is a leap year. A year is a leap year if it is divisible by 4 but not by 100, or it is divisible by 400.
- `boolean isValidDate(int year, int month, int day)`: returns true if the given year, month and day constitute a given date. Assume that year is between 1 and 9999, month is

between 1 (Jan) to 12 (Dec) and day shall be between 1 and 28|29|30|31 depending on the month and whether it is a leap year.

- `int getDayOfWeek(int year, int month, int day)`: returns the day of the week, where 0 for SUN, 1 for MON, ..., 6 for SAT, for the given date. Assume that the date is valid.
- `String toString(int year, int month, int day)`: prints the given date in the format "xxxday d mmm yyyy", e.g., "Tuesday 14 Feb 2012". Assume that the given date is valid.

Hints

To find the day of the week (Reference: Wiki "Determination of the day of the week"):

1700-	1800-	1900-	2000-	2100-	2200-	2300-	2400-
4	2	0	6	4	2	0	6

1. Based on the first two digit of the year, get the number from the following "century" table.
2. Take note that the entries 4, 2, 0, 6 repeat.
3. Add to the last two digit of the year.
4. Add to "the last two digit of the year divide by 4, truncate the fractional part".
5. Add to the number obtained from the following month table:

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Non-Leap Year	0	3	3	6	1	4	6	2	5	0	3	5
Leap Year	6	2	same as above									

6. Add to the day.
7. The sum modulus 7 gives the day of the week, where 0 for SUN, 1 for MON, ..., 6 for SAT.

For example: 2012, Feb, 17

$(6 + 12 + 12/4 + 2 + 17) \% 7 = 5$ (Fri)

The skeleton of the program is as follows:

```
/* Utilities for Date Manipulation */
public class DateUtil {

    // Month's name – for printing
    public static String[] strMonths
        = { "Jan", "Feb", "Mar", "Apr", "May", "Jun",
            "Jul", "Aug", "Sep", "Oct", "Nov", "Dec" };

    // Number of days in each month (for non-leap years)
    public static int[] daysInMonths
        = { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };

    // Returns true if the given year is a leap year
    public static boolean isLeapYear(int year) { ..... }

    // Return true if the given year, month, day is a valid date
    // year: 1-9999
    // month: 1(Jan)-12(Dec)
    // day: 1-28|29|30|31. The last day depends on year and month
    public static boolean isValidDate(int year, int month, int day) { ..... }

    // Return the day of the week, 0:Sun, 1:Mon, ..., 6:Sat
    public static int getDayOfWeek(int year, int month, int day) { ..... }
```

```
// Return String "xxxday d mmm yyyy" (e.g., Wednesday 29 Feb 2012)
public static String printDate(int year, int month, int day) { ..... }

// Test Driver
public static void main(String[] args) {
    System.out.println(isLeapYear(1900)); // false
    System.out.println(isLeapYear(2000)); // true
    System.out.println(isLeapYear(2011)); // false
    System.out.println(isLeapYear(2012)); // true

    System.out.println(isValidDate(2012, 2, 29)); // true
    System.out.println(isValidDate(2011, 2, 29)); // false
    System.out.println(isValidDate(2099, 12, 31)); // true
    System.out.println(isValidDate(2099, 12, 32)); // false

    System.out.println(getDayOfWeek(1982, 4, 24)); // 6:Sat
    System.out.println(getDayOfWeek(2000, 1, 1)); // 6:Sat
    System.out.println(getDayOfWeek(2054, 6, 19)); // 5:Fri
    System.out.println(getDayOfWeek(2012, 2, 17)); // 5:Fri

    System.out.println(toString(2012, 2, 14)); // Tuesday 14 Feb 2012
}
}
```

Notes

You can compare the day obtained with the Java's Calendar class as follows:

```
// Construct a Calendar instance with the given year, month and day
Calendar cal = new GregorianCalendar(year, month - 1, day); // month is 0-based
// Get the day of the week number: 1 (Sunday) to 7 (Saturday)
int dayNumber = cal.get(Calendar.DAY_OF_WEEK);
String[] calendarDays = { "Sunday", "Monday", "Tuesday", "Wednesday",
    "Thursday", "Friday", "Saturday" };

// Print result
System.out.println("It is " + calendarDays[dayNumber - 1]);
```

The calendar we used today is known as Gregorian calendar, which came into effect in October 15, 1582 in some countries and later in other countries. It replaces the Julian calendar. 10 days were removed from the calendar, i.e., October 4, 1582 (Julian) was followed by October 15, 1582 (Gregorian). The only difference between the Gregorian and the Julian calendar is the "leap-year rule". In Julian calendar, every four years is a leap year. In Gregorian calendar, a leap year is a year that is divisible by 4 but not divisible by 100, or it is divisible by 400, i.e., the Gregorian calendar omits century years which are not divisible by 400. Furthermore, Julian calendar considers the first day of the year as march 25th, instead of January 1st.

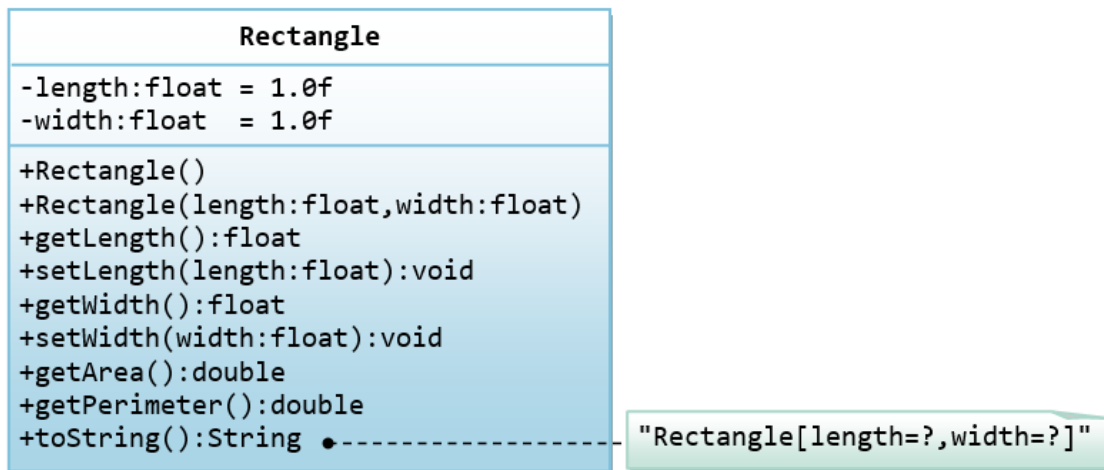
This above algorithm work for Gregorian dates only. It is difficult to modify the above algorithm to handle pre-Gregorian dates. A better algorithm is to find the number of days from a known date.

12. Exercises on Classes and Objects

12.1 The Rectangle Class

A class called Rectangle, which models a rectangle with a length and a width (in float), is designed as shown in the following class diagram. Write the Rectangle class.

Hints:



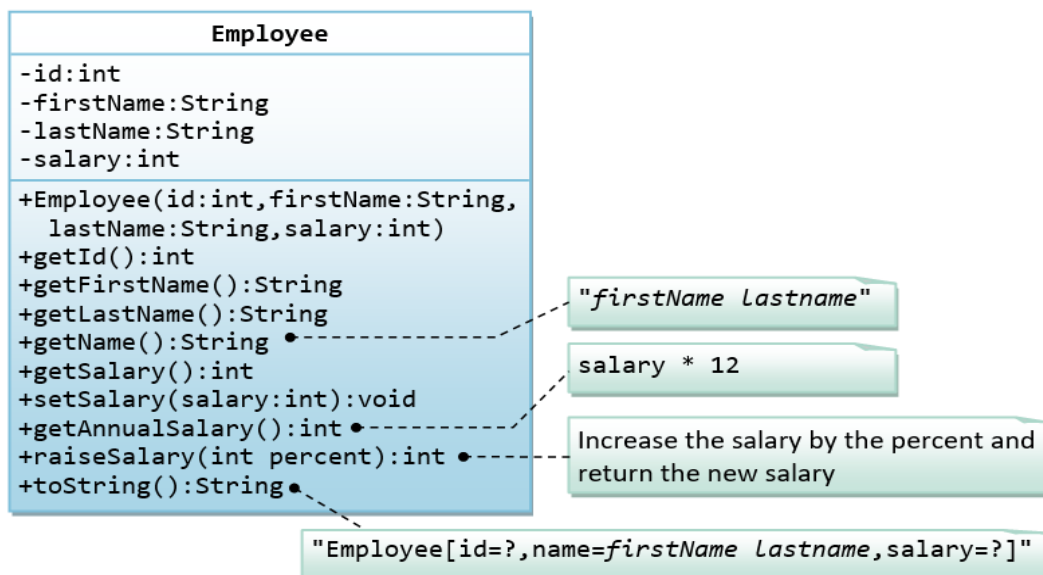
The expected output is:

```
Rectangle[length=1.2,width=3.4]
Rectangle[length=1.0,width=1.0]
Rectangle[length=5.6,width=7.8]
length is: 5.6
width is: 7.8
area is: 43.68
perimeter is: 26.80
```

12.2 The Employee Class

A class called Employee, which models an employee with an ID, name and salary, is designed as shown in the following class diagram. The method `raiseSalary(percent)` increases the salary by the given percentage. Write the Employee class.

Hints:



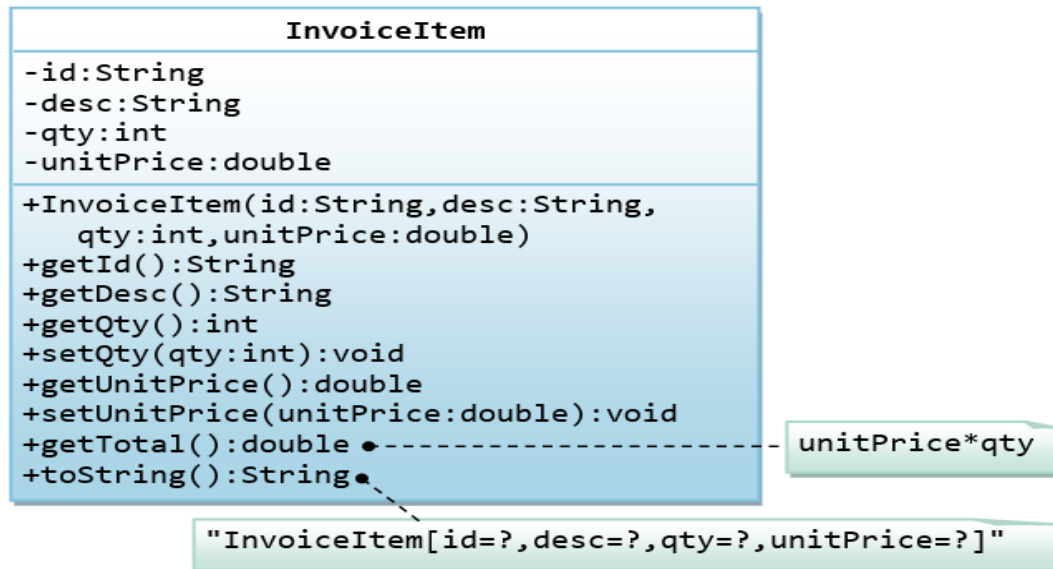
The expected out is:

```
Employee[id=8,name=Peter Tan,salary=2500]
Employee[id=8,name=Peter Tan,salary=999]
id is: 8
firstname is: Peter
lastname is: Tan
salary is: 999
name is: Peter Tan
annual salary is: 11988
```


12.3 The InvoiceItem Class

A class called InvoiceItem, which models an item of an invoice, with ID, description, quantity and unit price, is designed as shown in the following class diagram. Write the InvoiceItem class.

Hints:



The expected output is:

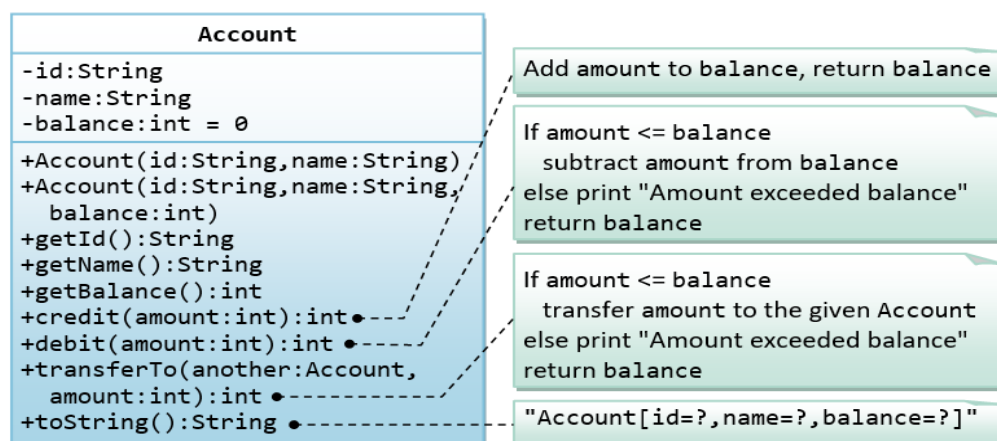
```

InvoiceItem[id=A101,desc=Pen Red,qty=888,unitPrice=0.08]
InvoiceItem[id=A101,desc=Pen Red,qty=999,unitPrice=0.99]
id is: A101
desc is: Pen Red
qty is: 999
unitPrice is: 0.99
The total is: 989.01
  
```

12.4 The Account Class

A class called Account, which models a bank account of a customer, is designed as shown in the following class diagram. The methods credit(amount) and debit(amount) add or subtract the given amount to the balance. The method transferTo(anotherAccount, amount) transfers the given amount from this Account to the given anotherAccount. Write the Account class.

Hints:



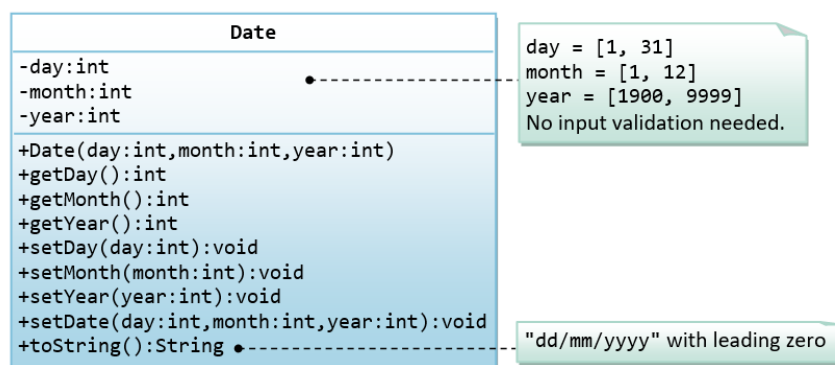
The expected output is:

```
Account[id=A101,name=Tan Ah Teck,balance=88]
Account[id=A102,name=Kumar,balance=0]
ID: A101
Name: Tan Ah Teck
Balance: 88
Account[id=A101,name=Tan Ah Teck,balance=188]
Account[id=A101,name=Tan Ah Teck,balance=138]
Amount exceeded balance
Account[id=A101,name=Tan Ah Teck,balance=138]
Account[id=A101,name=Tan Ah Teck,balance=38]
Account[id=A102,name=Kumar,balance=100]
```

12.5 The Date Class

A class called Date, which models a calendar date, is designed as shown in the following class diagram. Write the Date class.

Hints:



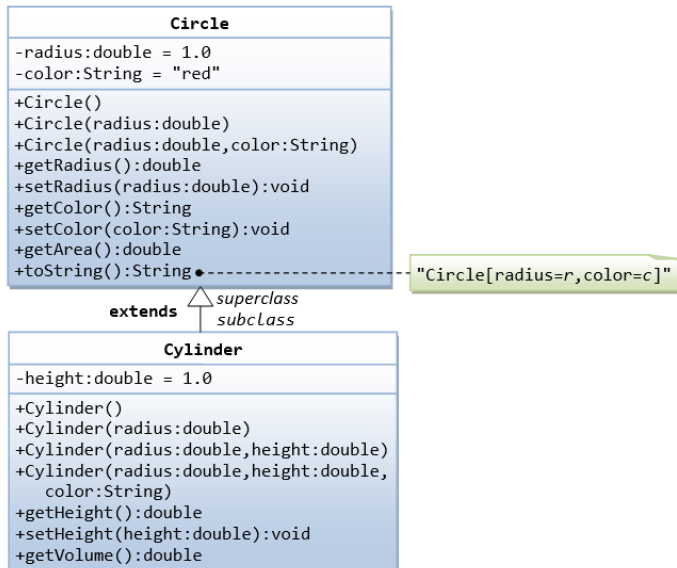
The expected output is:

```
01/02/2014
09/12/2099
Month: 12
Day: 9
Year: 2099
03/04/2016
```

13. Exercises on Inheritance

13.1 An Introduction to OOP Inheritance - The Circle and Cylinder Classes

This exercise shall guide you through the important concepts in inheritance.



In this exercise, a subclass called Cylinder is derived from the superclass Circle as shown in the class diagram (where an arrow pointing up from the subclass to its superclass). Study how the subclass Cylinder invokes the superclass' constructors (via `super()` and `super(radius)`) and inherits the variables and methods from the superclass Circle.

You can reuse the Circle class that you have created in the previous exercise. Make sure that you keep "Circle.class" in the same directory.

```

public class Cylinder extends Circle { // Save as "Cylinder.java"
    private double height; // private variable

    // Constructor with default color, radius and height
    public Cylinder() {
        super(); // call superclass no-arg constructor Circle()
        height = 1.0;
    }

    // Constructor with default radius, color but given height
    public Cylinder(double height) {
        super(); // call superclass no-arg constructor Circle()
        this.height = height;
    }

    // Constructor with default color, but given radius, height
    public Cylinder(double radius, double height) {
        super(radius); // call superclass constructor Circle(r)
        this.height = height;
    }

    // A public method for retrieving the height
    public double getHeight() {
        return height;
    }

    // A public method for computing the volume of cylinder
    // use superclass method getArea() to get the base area
    public double getVolume() {
        return getArea()*height;
    }
}

```

Method Overriding and "Super": The subclass Cylinder inherits `getArea()` method from its superclass Circle. Try overriding the `getArea()` method in the subclass Cylinder to compute the surface area ($=2\pi \times \text{radius} \times \text{height} + 2 \times \text{base-area}$) of the cylinder instead of base area. That is, if `getArea()` is called by

a Circle instance, it returns the area. If getArea() is called by a Cylinder instance, it returns the surface area of the cylinder.

If you override the getArea() in the subclass Cylinder, the getVolume() no longer works. This is because the getVolume() uses the overridden getArea() method found in the same class. (Java runtime will search the superclass only if it cannot locate the method in this class). Fix the getVolume().

Hints: After overriding the getArea() in subclass Cylinder, you can choose to invoke the getArea() of the superclass Circle by calling super.getArea().

Try:

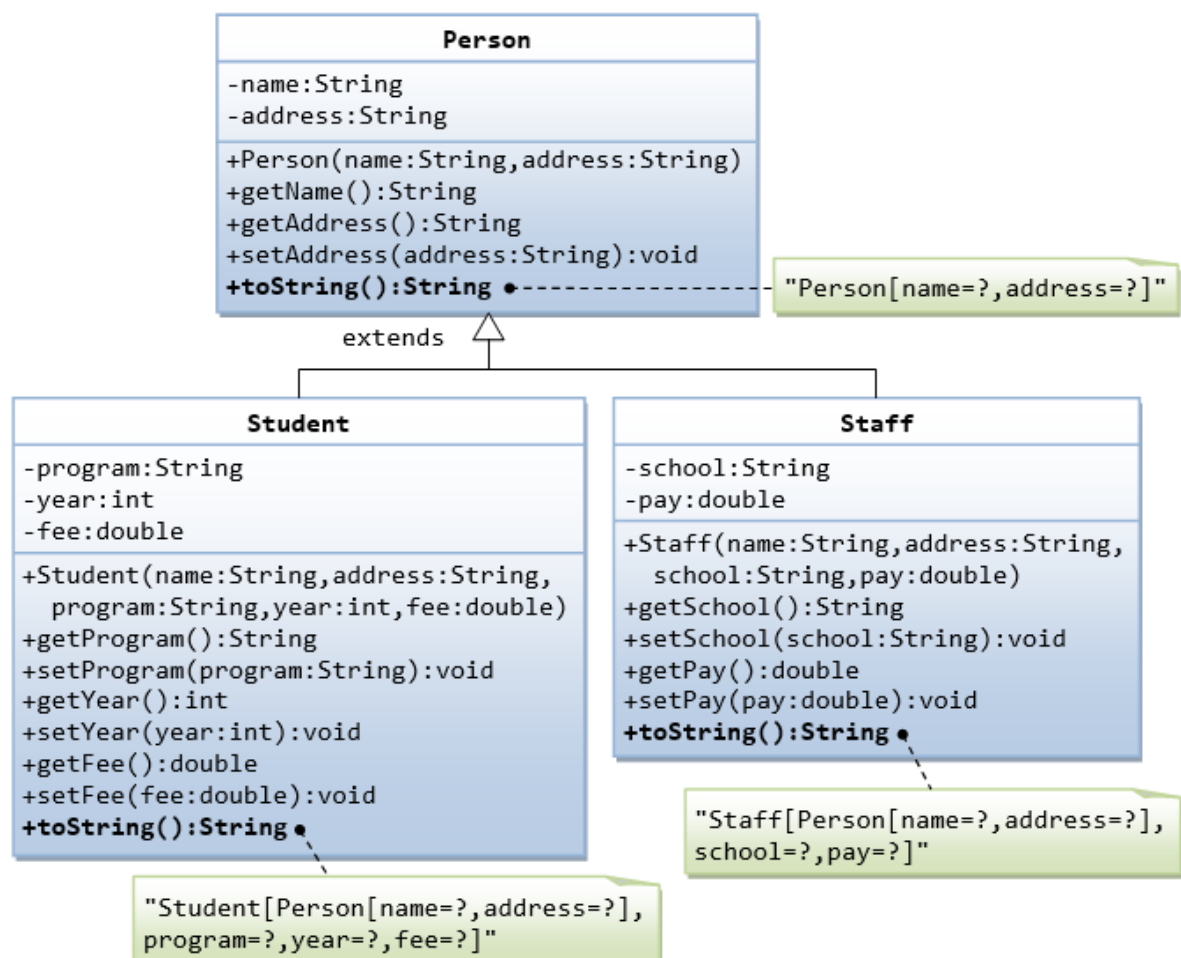
Provide a toString() method to the Cylinder class, which overrides the toString() inherited from the superclass Circle, e.g.,

```
@Override
public String toString() {    // in Cylinder class
    return "Cylinder: subclass of " + super.toString() // use Circle's toString()
        + " height=" + height;
}
```

Try out the toString() method in TestCylinder.

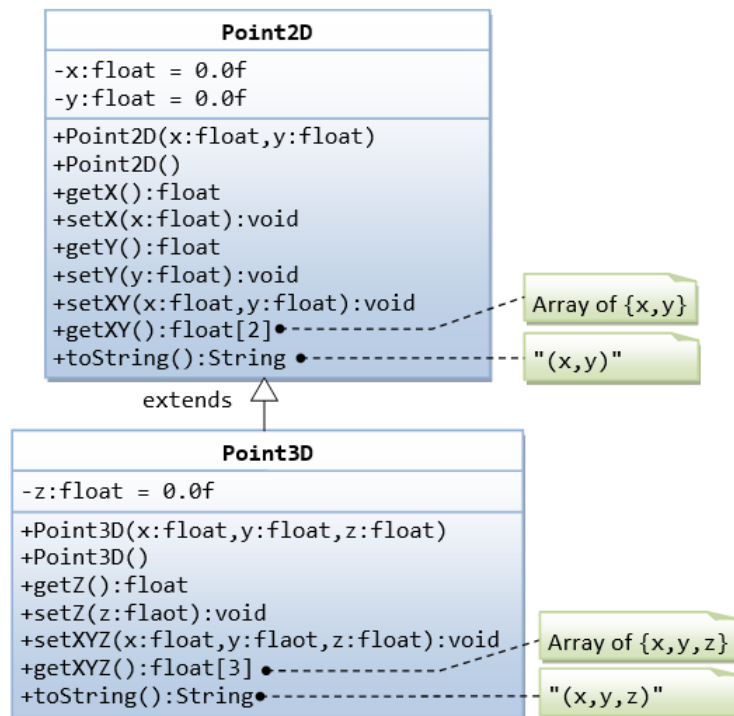
13.2 Superclass Person and its subclasses

Write the classes as shown in the following class diagram. Mark all the overridden methods with annotation @Override.



13.3 Point2D and Point3D

Write the classes as shown in the following class diagram. Mark all the overridden methods with annotation @Override.



Hints:

1. You cannot assign floating-point literal say 1.1 (which is a double) to a float variable, you need to add a suffix f, e.g. 0.0f, 1.1f.
2. The instance variables x and y are private in Point2D and cannot be accessed directly in the subclass Point3D. You need to access via the public getters and setters. For example,

```

public void setXYZ(float x, float y, float z) {
    setX(x); // or super.setX(x), use setter in superclass
    setY(y);
    this.z = z;
}

```

3. The method getXY() shall return a float array:

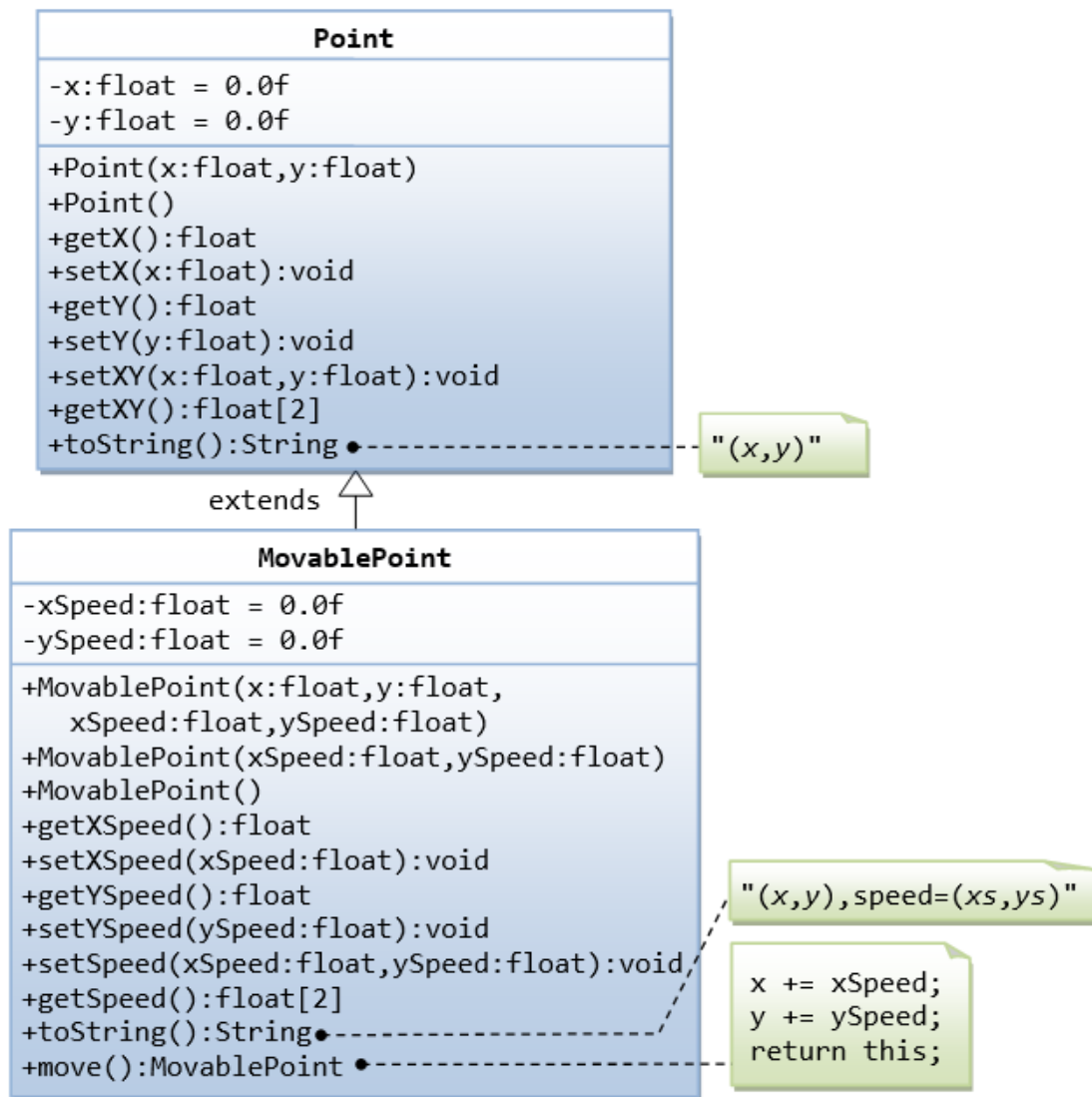
```

public float[] getXY() {
    float[] result = new float[2]; // construct an array of 2 elements
    result[0] = ...
    result[1] = ...
    return result; // return the array
}

```

13.4 Point and MovablePoint

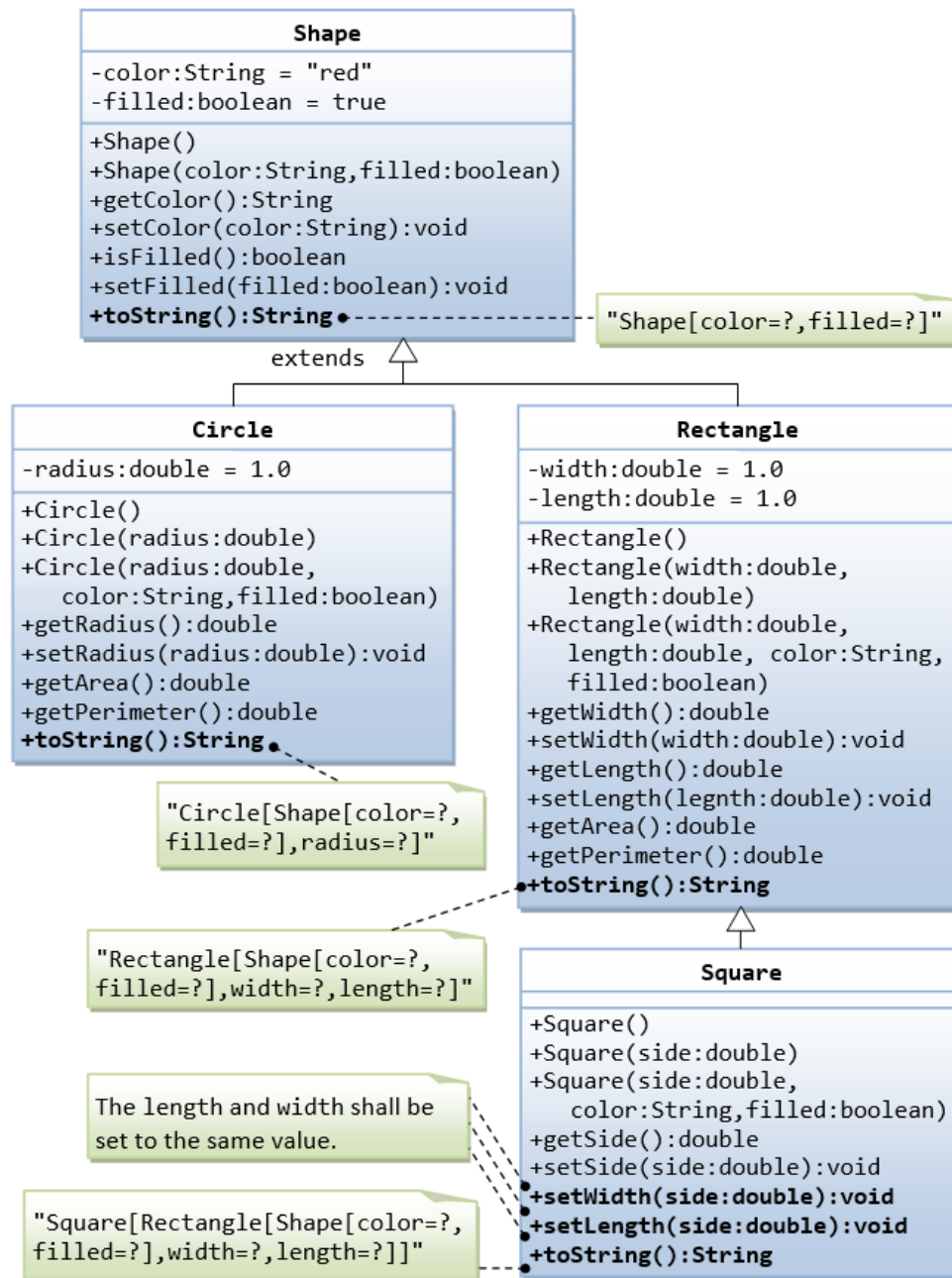
Write the classes as shown in the following class diagram. Mark all the overridden methods with annotation `@Override`.



Hints

1. You cannot assign floating-point literal say 1.1 (which is a double) to a float variable, you need to add a suffix f, e.g. 0.0f, 1.1f.
2. The instance variables x and y are private in Point and cannot be accessed directly in the subclass MovablePoint. You need to access via the public getters and setters. For example, you cannot write `x += xSpeed`, you need to write `setX(getX() + xSpeed)`.

13.5 Superclass **Shape** and its subclasses **Circle**, **Rectangle** and **Square**



Write a superclass called Shape (as shown in the class diagram)

Write a test program to test all the methods defined in Shape.

Write two subclasses of Shape called Circle and Rectangle, as shown in the class diagram.

Write a class called Square, as a subclass of Rectangle. Convince yourself that Square can be modeled as a subclass of Rectangle. Square has no instance variable, but inherits the instance variables width and length from its superclass Rectangle.

- Provide the appropriate constructors (as shown in the class diagram).

Hints:

```

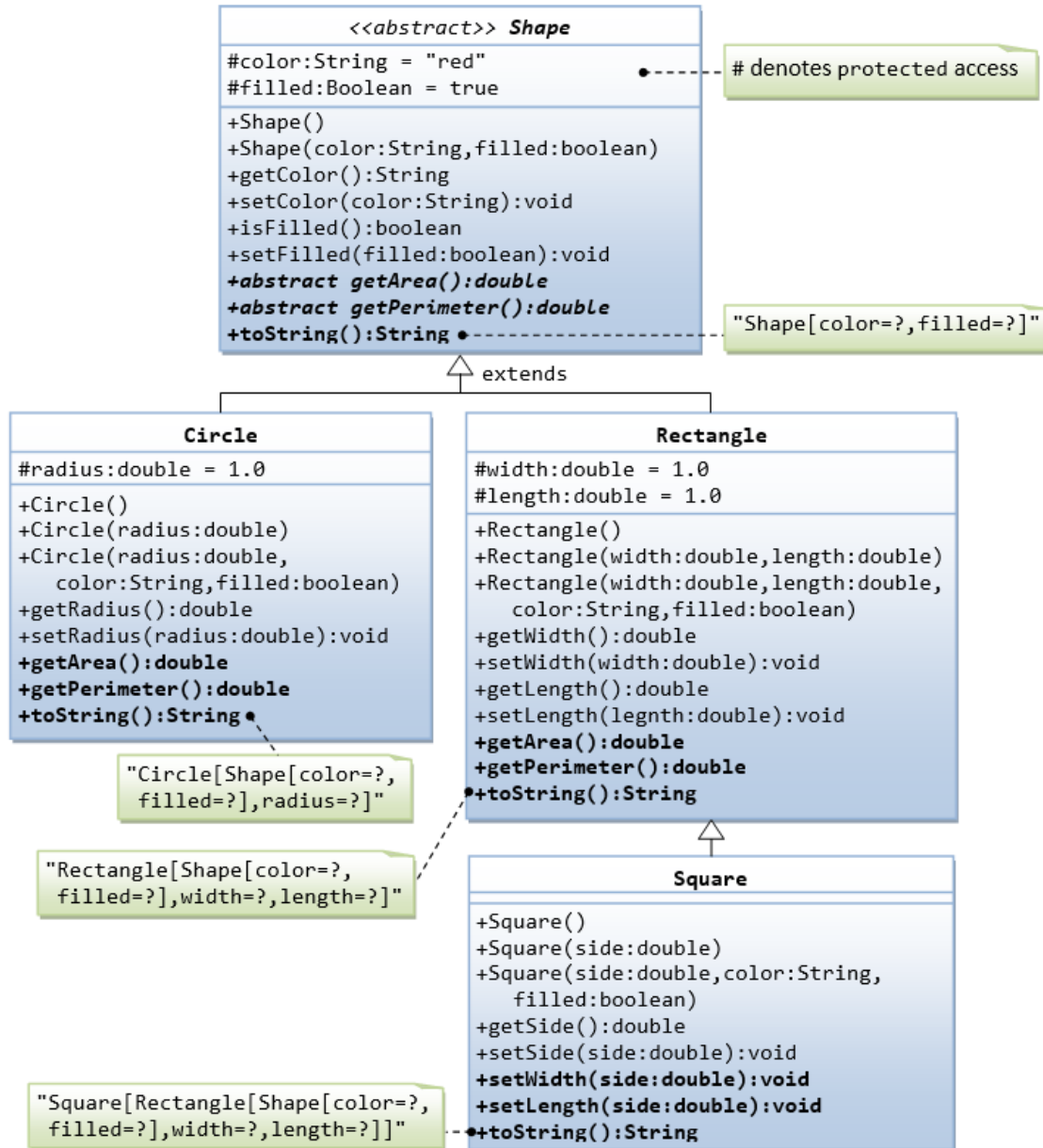
public Square(double side) {
    super(side, side); // Call superclass Rectangle(double, double)
}

```

- Override the toString() method to return "A Square with side=xxx, which is a subclass of yyy", where yyy is the output of the toString() method from the superclass.
- Do you need to override the getArea() and getPerimeter()? Try them out.
- Override the setLength() and setWidth() to change both the width and length, so as to maintain the square geometry.

14.1 Ex: Abstract Superclass **Shape** and Its Concrete Subclasses

Rewrite the superclass Shape and its subclasses Circle, Rectangle and Square, as shown in the class diagram. Shape is an abstract class containing 2 abstract methods: `getArea()` and `getPerimeter()`, where its concrete subclasses must provide its implementation. All instance variables shall have protected access, i.e., accessible by its subclasses and classes in the same package. Mark all the overridden methods with annotation `@Override`.



In this exercise, Shape shall be defined as an abstract class, which contains:

- Two protected instance variables **color(String)** and **filled(boolean)**. The protected variables can be accessed by its subclasses and classes in the same package. They are denoted with a '#' sign in the class diagram.
- Getter and setter for all the instance variables, and `toString()`.
- Two abstract methods `getArea()` and `getPerimeter()` (shown in italics in the class diagram).
- Subclasses Circle and Rectangle shall override the abstract methods `getArea()` and `getPerimeter()` and provide the proper implementation. They also override the `toString()`.

Write a test class to test these statements involving polymorphism and explain the outputs. Some statements may trigger compilation errors. Explain the errors, if any.


```

Shape s1 = new Circle(5.5, "red", false); // Upcast Circle to Shape
System.out.println(s1);                // which version?
System.out.println(s1.getArea());       // which version?
System.out.println(s1.getPerimeter());  // which version?
System.out.println(s1.getColor());
System.out.println(s1.isFilled());
System.out.println(s1.getRadius());

Circle c1 = (Circle)s1;                 // Downcast back to Circle
System.out.println(c1);
System.out.println(c1.getArea());
System.out.println(c1.getPerimeter());
System.out.println(c1.getColor());
System.out.println(c1.isFilled());
System.out.println(c1.getRadius());

Shape s2 = new Shape();

Shape s3 = new Rectangle(1.0, 2.0, "red", false); // Upcast
System.out.println(s3);
System.out.println(s3.getArea());
System.out.println(s3.getPerimeter());
System.out.println(s3.getColor());
System.out.println(s3.getLength());

Rectangle r1 = (Rectangle)s3; // downcast
System.out.println(r1);
System.out.println(r1.getArea());
System.out.println(r1.getColor());
System.out.println(r1.getLength());

Shape s4 = new Square(6.6); // Upcast
System.out.println(s4);
System.out.println(s4.getArea());
System.out.println(s4.getColor());
System.out.println(s4.getSide());

// Take note that we downcast Shape s4 to Rectangle,
// which is a superclass of Square, instead of Square
Rectangle r2 = (Rectangle)s4;
System.out.println(r2);
System.out.println(r2.getArea());
System.out.println(r2.getColor());
System.out.println(r2.getSide());
System.out.println(r2.getLength());

// Downcast Rectangle r2 to Square
Square sq1 = (Square)r2;
System.out.println(sq1);
System.out.println(sq1.getArea());
System.out.println(sq1.getColor());
System.out.println(sq1.getSide());
System.out.println(sq1.getLength());

```

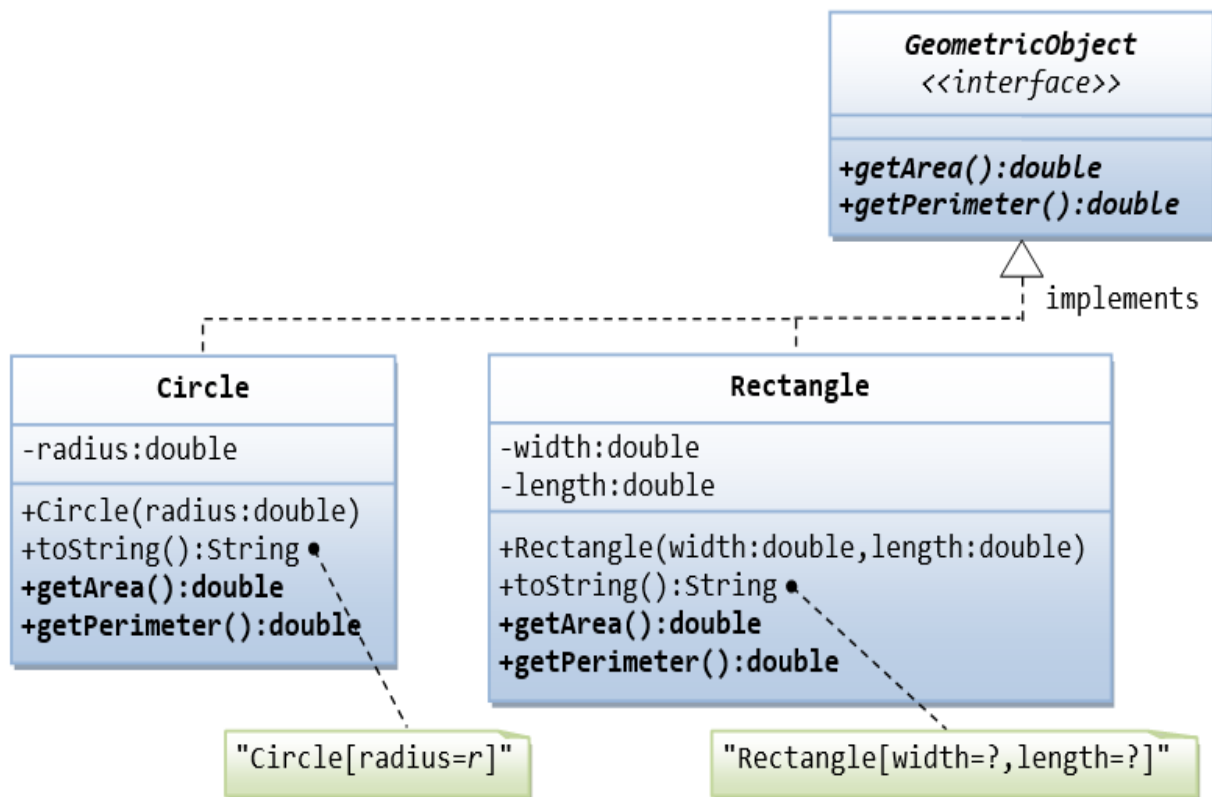
Try:

Explain the usage of the abstract method and abstract class?

14.2 GeometricObject Interface and its Implementation

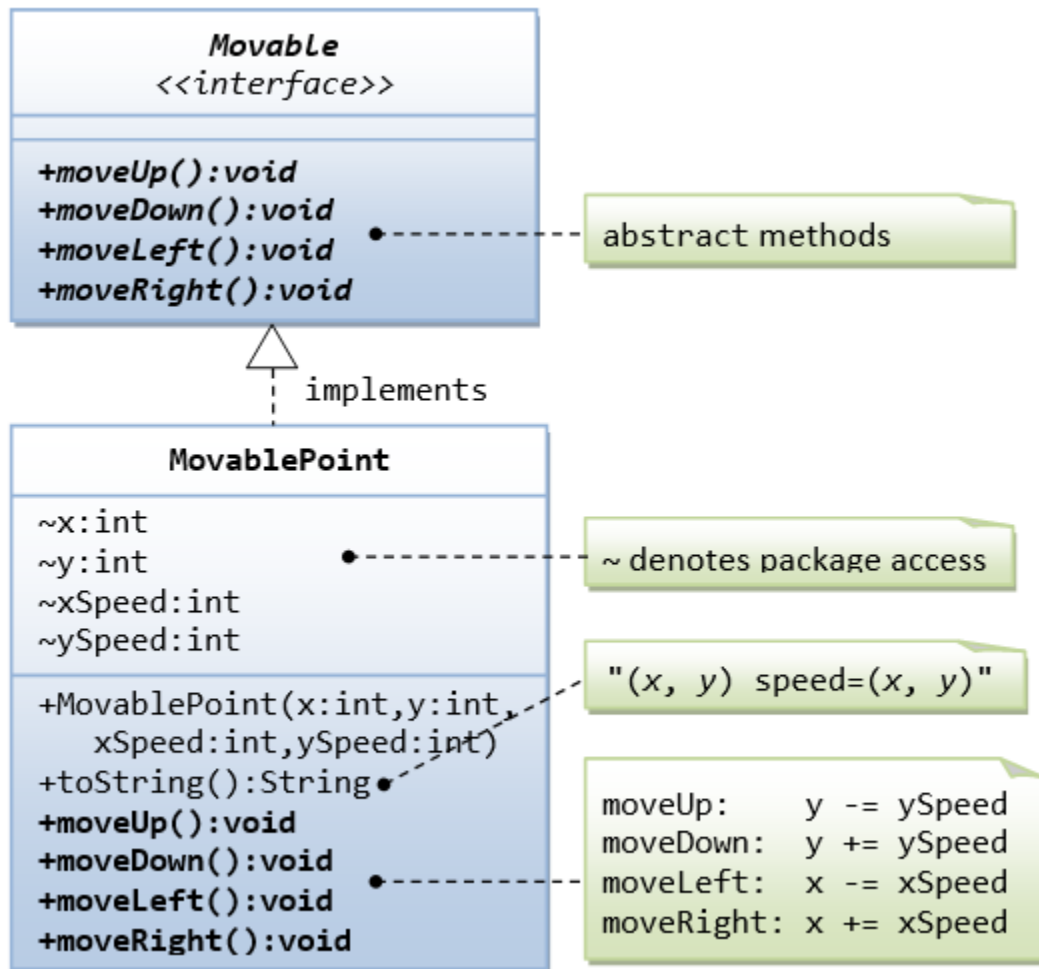
Classes Circle and Rectangle

Write an interface called GeometricObject, which contains 2 abstract methods: `getArea()` and `getPerimeter()`, as shown in the class diagram. Also write an implementation class called Circle. Mark all the overridden methods with annotation `@Override`.



14.3 Ex: Movable Interface and its Implementation MovablePoint Class

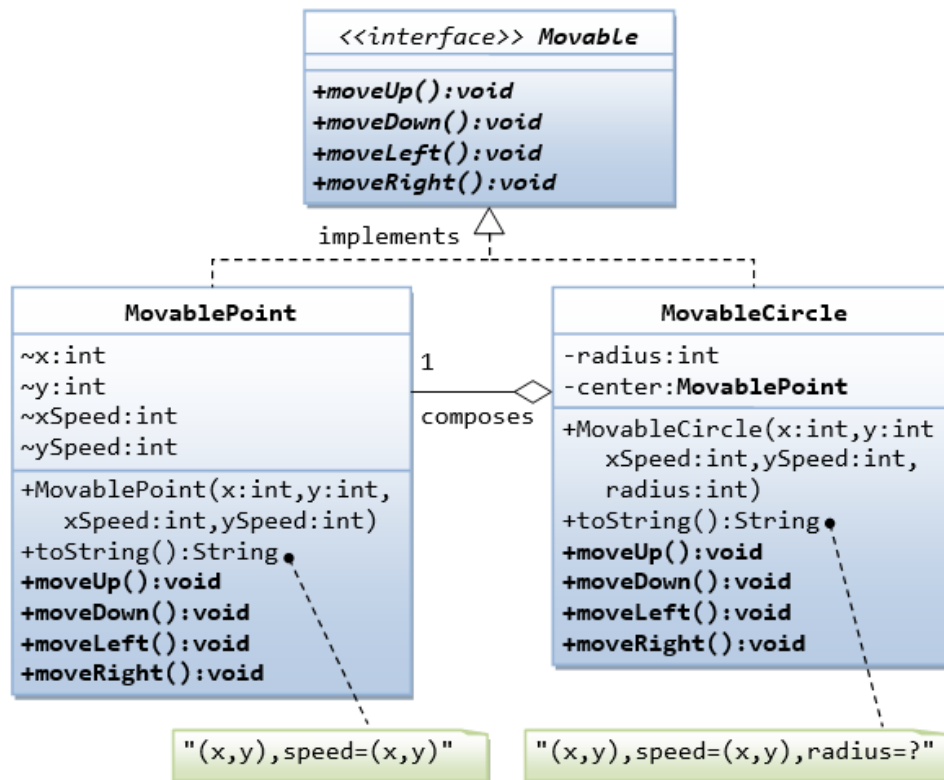
Write an interface called Movable, which contains 4 abstract methods moveUp(), moveDown(), moveLeft() and moveRight(), as shown in the class diagram. Also write an implementation class called MovablePoint. Mark all the overridden methods with annotation @Override.



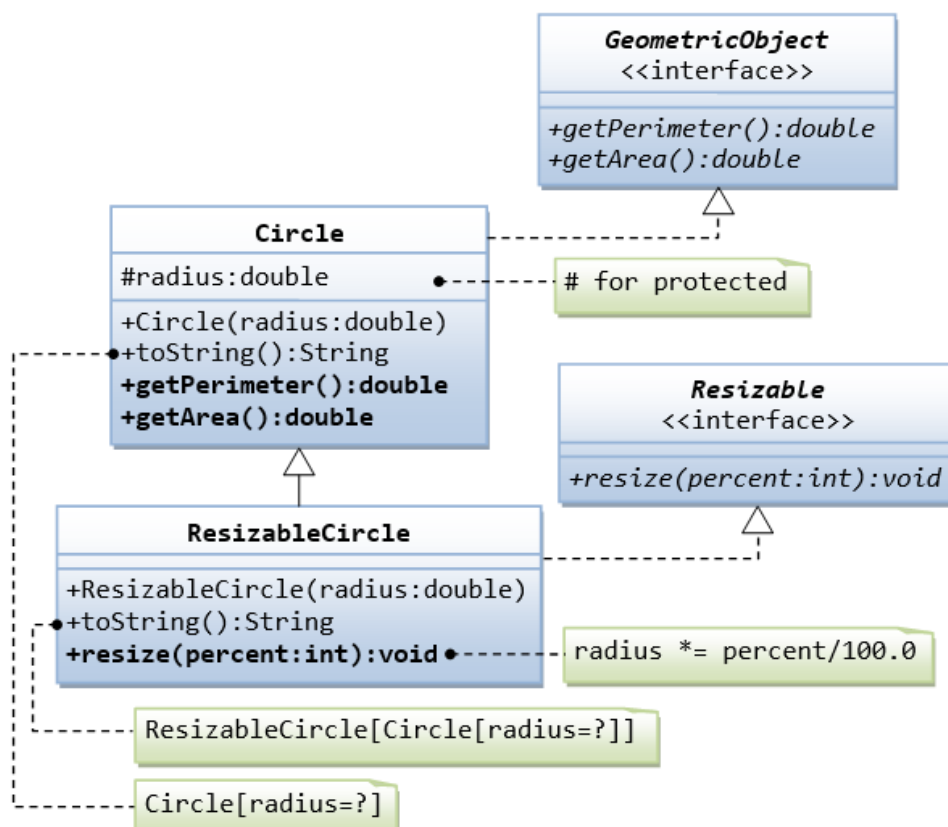
14.4 Movable Interface and Classes MovablePoint and MovableCircle

Write an interface called Movable, which contains 4 abstract methods moveUp(), moveDown(), moveLeft() and moveRight(), as shown in the class diagram.

Write the implementation classes called MovablePoint and MovableCircle. Mark all the overridden methods with annotation @Override.



14.5 Interfaces **Resizable** and **GeometricObject**



Write the interface called **GeometricObject**, which declares two abstract methods: `getParameter()` and `getArea()`, as specified in the class diagram.

Hints:

```
public interface GeometricObject {
    public double getPerimeter();
    .....
}
```

Write the implementation class Circle, with a protected variable radius, which implements the interface GeometricObject.

Hints:

```
public class Circle implements GeometricObject {
    // Private variable
    .....

    // Constructor
    .....

    // Implement methods defined in the interface GeometricObject
    @Override
    public double getPerimeter() { ..... }

    .....
}
```

Write a test program called TestCircle to test the methods defined in Circle.

The class ResizableCircle is defined as a subclass of the class Circle, which also implements an interface called Resizable, as shown in class diagram. The interface Resizable declares an abstract method resize(), which modifies the dimension (such as radius) by the given percentage. Write the interface Resizable and the class ResizableCircle.

Hints:

```
public interface Resizable {
    public double resize(...);
}
```

```
public class ResizableCircle extends Circle implements Resizable {

    // Constructor
    public ResizableCircle(double radius) {
        super(...);
    }

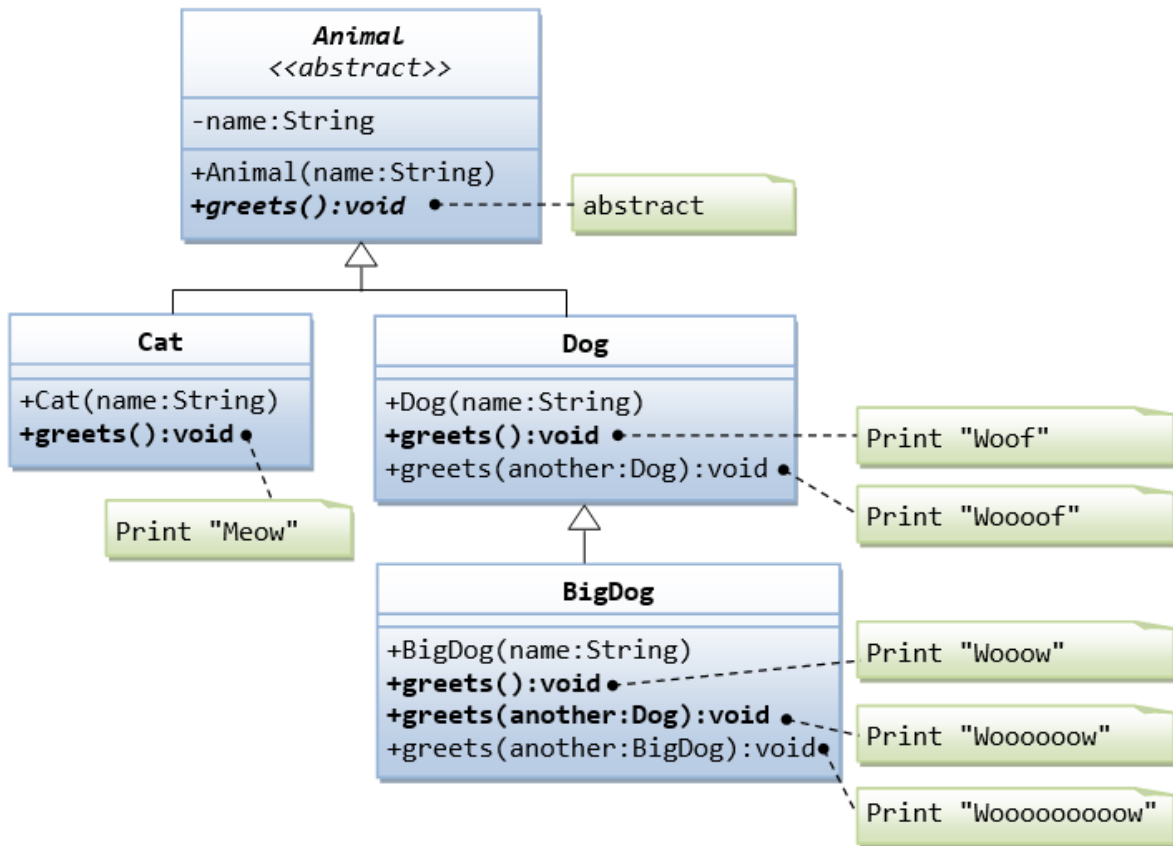
    // Implement methods defined in the interface Resizable
    @Override
    public double resize(int percent) { ..... }
}
```

Try:

Write a test program called TestResizableCircle to test the methods defined in ResizableCircle.

14.6 Abstract Superclass **Animal** and its Implementation Subclasses

Write the codes for all the classes shown in the class diagram. Mark all the overridden methods with annotation @Override.



14.7 Another View of Abstract Superclass **Animal** and its Implementation Subclasses

Examine the following codes and draw the class diagram.

```

abstract public class Animal {
    abstract public void greeting();
}

```

```

public class Cat extends Animal {
    @Override
    public void greeting() {
        System.out.println("Meow!");
    }
}

public class Dog extends Animal {
    @Override
    public void greeting() {
        System.out.println("Woof!");
    }

    public void greeting(Dog another) {
        System.out.println("Woowoooooof!");
    }
}

```

```

public class BigDog extends Dog {
    @Override
    public void greeting() {
        System.out.println("Woow!");
    }

    @Override
    public void greeting(Dog another) {

```

```

    System.out.println("Woooooowwww!");
}
}

```

Try:

Explain the outputs (or error) for the following test program.

```

public class TestAnimal {
    public static void main(String[] args) {
        // Using the subclasses
        Cat cat1 = new Cat();
        cat1.greeting();
        Dog dog1 = new Dog();
        dog1.greeting();
        BigDog bigDog1 = new BigDog();
        bigDog1.greeting();

        // Using Polymorphism
        Animal animal1 = new Cat();
        animal1.greeting();
        Animal animal2 = new Dog();
        animal2.greeting();
        Animal animal3 = new BigDog();
        animal3.greeting();
        Animal animal4 = new Animal();

        // Downcast
        Dog dog2 = (Dog)animal2;
        BigDog bigDog2 = (BigDog)animal3;
        Dog dog3 = (Dog)animal3;
        Cat cat2 = (Cat)animal2;
        dog2.greeting(dog3);
        dog3.greeting(dog2);
        dog2.greeting(bigDog2);
        bigDog2.greeting(dog2);
        bigDog2.greeting(bigDog1);
    }
}

```

15. Final Notes

The only way to learn programming is program, program and program on challenging problems. The problems in this tutorial are certainly NOT challenging. There are tens of thousands of challenging problems available – used in training for various programming contests (such as International Collegiate Programming Contest (ICPC), International Olympiad in Informatics (IOI)). Check out these sites:

- The ACM - ICPC International collegiate programming contest (<https://icpc.global/>)
- The Topcoder Open (TCO) annual programming and design contest (<https://www.topcoder.com/>)
- Universidad de Valladolid's online judge (<https://uva.onlinejudge.org/>).
- Peking University's online judge (<http://poj.org/>).
- USA Computing Olympiad (USACO) Training Program @ <http://train.usaco.org/usacogate>.
- Google's coding competitions (<https://codingcompetitions.withgoogle.com/codejam>,
<https://codingcompetitions.withgoogle.com/hashcode>)
- The ICFP programming contest (<https://www.icfpconference.org/>)
- BME International 24-hours programming contest (<https://www.challenge24.org/>)
- The International Obfuscated C Code Contest (<https://www0.us.ioccc.org/main.html>)
- Internet Problem Solving Contest (<https://ipsc.ksp.sk/>)
- Microsoft Imagine Cup (<https://imaginecup.microsoft.com/en-us>)
- Hewlett Packard Enterprise (HPE) Codewars (<https://hpecodewars.org/>)
- OpenChallenge (<https://www.openchallenge.org/>)

Coding Contests Scores

Students must solve problems and attain scores in the following coding contests:

Name of the contest	Minimum number of problems to solve	Required score
• CodeChef	20	200
• Leetcode	20	200
• GeeksforGeeks	20	200
• SPOJ	5	50
• InterviewBit	10	1000
• Hackerrank	25	250
• Codeforces	10	100
• BuildIT	50	500
Total score need to obtain		2500

Student must have any one of the following certifications:

- HackerRank – Java Basic Skills Certification
- Oracle Certified Associate Java Programmer OCAJP
- CodeChef - Learn Java Certification
- NPTEL – Programming in Java
- NPTEL – Data Structures and Algorithms in Java

V. TEXT BOOKS:

1. Farrell, Joyce. Java Programming, Cengage Learning B S Publishers, 8th Edition, 2020
2. Schildt, Herbert. Java: The Complete Reference 11th Edition, McGraw-Hill Education, 2018.

VI. REFERENCE BOOKS:

1. Deitel, Paul and Deitel, Harvey. Java: How to Program, Pearson, 11th Edition, 2018.
2. Evans, Benjamin J. and Flanagan, David. Java in a Nutshell, O'Reilly Media, 7th Edition, 2018.
3. Bloch, Joshua. Effective Java, Addison-Wesley Professional, 3rd Edition, 2017.
4. Sierra, Kathy and Bates, Bert. Head First Java, O'Reilly Media, 2nd Edition, 2005

VII. ELECTRONIC RESOURCES:

1. <https://docs.oracle.com/en/java/>
2. <https://www.geeksforgeeks.org/java>
3. <https://www.tutorialspoint.com/java/index.htm>
4. <https://www.coursera.org/courses?query=java>

VIII. MATERIALS ONLINE;

1. Syllabus
2. Lab manual

COURSE CONTENT

MANUFACTURING PRACTICE								
I Semester: CSE (AI & ML) / IT / ECE / EEE								
II Semester: CSE / CSE (DS) / CSE (CS)								
Course Code	Category	Hours / Week			Credits	Maximum Marks		
AMED02	Foundation	L	T	P	C	CIA	SEE	Total
		-	1	2	2	40	60	100
Contact Classes: Nil	Tutorial Classes: 15	Practical Classes:30			Total Classes:45			
Prerequisite: There is no prerequisite for this course.								

I. COURSE OVERVIEW:

This course provides the opportunity to become confident with new tools, equipment, and techniques for creating physical objects and mechanisms with a variety of materials. The students will learn the concepts of 3D printing, laser cutting, circuit board soldering, wood carving and CNC machining. Skills learned in the course enable the students about the design process in digital manufacturing used in various industrial applications.

II. COURSES OBJECTIVES:

The students will try to learn

- I. The digital and additive manufacturing techniques used in various industrial applications in the current era to develop prototype models.
- II. The unconventional machining processes and their selective applications as an alternative to traditional manufacturing methods.
- III. The standard electrical wiring practices for domestic and industrial appliances.
- IV. The soldering and de-soldering components on a circuit board safely and correctly.

III. COURSE OUTCOMES:

At the end of the course students should be able to:

- CO 1 Practice the various types of manufacturing methods for preparing the given material to desired shape by using traditional and unconventional manufacturing practices.
- CO 2 Execute the additive manufacturing technology for learning about the 3D printing processes and techniques.
- CO 3 Select computer numerical control laser techniques for preparing the required geometrical profiles on non-metallic materials.
- CO 4 Demonstrate the assembly and disassembly of electrical equipment's and controls for safe domestic applications.
- CO 5 Make use of computer numerical technologies to create products using wood carving techniques.
- CO 6 Apply the plumbing skills to work with fittings and pipes made of PVC and galvanized steel.

IV. COURSE CONTENT:

EXERCISES ON MANUFACTURING PRACTICES

Note: Students are encouraged to bring their own laptops for laboratory practice sessions.

All dimensions are in mm in experiments.

Safety

Safety is a vital issue in all workplaces. Before using any equipment and machines or attempt practical work in a workshop everyone must understand basic safety rules. These rules will help keep all safe in the workshop.

Safety Rules

1. Always listen carefully to the teacher and follow instructions.
 2. When learning how to use a machine, listen very carefully to all the instructions given by the faculty / instructor. Ask questions, especially if you do not fully understand.
 3. Always wear an apron as it will protect your clothes and holds loose clothing such as ties in place.
 4. Wear good strong shoes.
 5. Bags should not be brought into a workshop as people can trip over them.
 6. Do not use a machine if you have not been shown how to operate it safely by the faculty / instructors
 7. Know where the emergency stop buttons are positioned in the workshop. If you see an accident at the other side of the workshop you can use the emergency stop button to turn off all electrical power to machines.
 8. Wherever required, wear protective equipment, such as goggles, safety glasses, masks, gloves, hair nets, etc.
 9. Always be patient, never rush in the workshop.
 10. Always use a guard when working on a machine.
 11. Keep hands away from moving/rotating machinery.
 12. Use hand tools carefully, keeping both hands behind the cutting edge.
 13. Report any UNSAFE condition or acts to instructor.
 14. Report any damage to machines/equipment as this could cause an accident.
 15. Keep your work area clean.
-

1. Getting Started Exercises

1.1 Principles of 3D printing and additive manufacturing techniques

3D printing or additive manufacturing enables to produce geometrically complex objects, shapes and textures. It often uses less material than traditional manufacturing methods and allows the production of prototypes / products that are not possible to produce economically with conventional manufacturing.

- Familiarization of 3D printing machine and its principle of operation.
- Standard use of Computer Aided Design (CAD) drawings .dwg format and Interface with CURA / Simplify 3D software as .stl file.
- Selection of Polylactic Acid (PLA) and Acrylonitrile Butadiene Styrene (ABS) materials and their specifications.

1.2 Preparation of stepped pulley with PLA material as shown in Figure 1.1

- Slicing of stepped pulley using .stl format
- Laying of stepped pulley using 3D printing

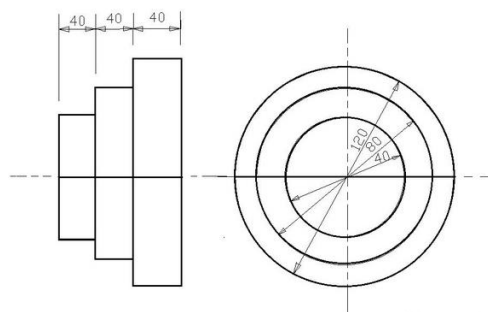


Figure 1.1 Stepped pulley

Try

- Preparation of spur gear with ABS material as shown in the Figure 1.2
 - Slicing of spur gear using .stl format
 - Laying of spur gear using 3D printing.

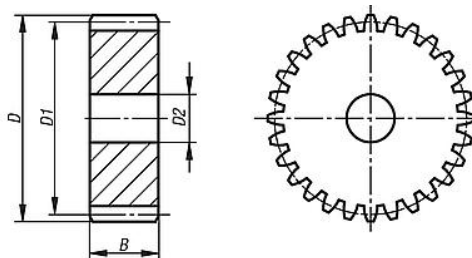


Figure 1.2 Spur gear

Dimensions:

$D = 40\text{mm}$

$D1 = 44\text{mm}$ Module, $m = 2\text{mm}$

$D2 = 20\text{mm}$ $B = 10\text{mm}$

Hint: Set the appropriate nozzle and temperature for ABS material, compare with PLA material.

2. Introduction to computerized numerical control (CNC) laser engraving

CNC Engraving machine is the process of gradually removing a small amount of material from a surface along a defined path. The process leaves a visible marking on the surface of the substrate (workpiece) with high accuracy and precision control.

- Familiarization of CNC engraving machine with action control.
- Create a visualization of the engraving pattern on CAD software. The student must have a solid grip on CAD to be able to create complex patterns in a quick and accurate manner.

- iii) Computer Aided Manufacturing (CAM) software is a special software that is used to generate programs for the CNC engraving machine.

2.1 Preparation of acrylic gears using CNC laser engraving / cutting machine as shown in Figure 2.1

- The parts file is divided into two pages. The file is a pdf. As different laser cutters use different file formats I leave it to you to convert it to a format suitable for your machine.
- All the parts apart for the dowel pieces cut out and ready to go. The blue piece is the clear acrylic front with its protective film still attached. Having cut out all the parts follow the instructions below to assemble the gear display.
- Glue together the two eighteen teeth gears. Make sure that they are aligned precisely.

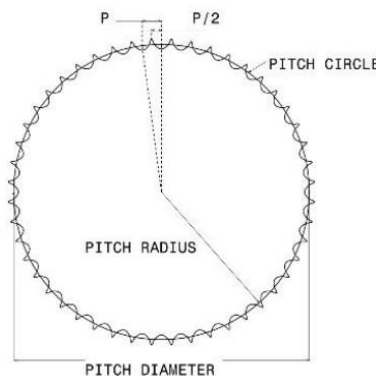


Figure 2.1 Acrylic gear

Try

- Preparation of artistic components IARE logo using CNC laser engraving as shown in the Figure 2.2

- Open your program and create new file (OPEN-NEW FILE), use the TEXT tool and write the word you want to engrave.
- Shows how to transform this word into an objects, each letter will become an individual shape.
- When using the laser technology, you have to make sure the file is one whole object, not a group of objects.



Figure 2.2 IARE logo

Hint: Set the appropriate parameters to perform operation on the balsaw wood material.

3. Introduction to computerized numerical control wood carving machines

CNC wood carving is the process of gradually removing a small amount of material from a surface along a defined path. The process leaves a visible marking on the surface of the substrate (workpiece) with high accuracy and precision control.

- Familiarization of CNC machine with action control.
- Create a visualization of the engraving pattern on CAD software. The student must have a solid grip on CAD to be able to create complex patterns in a quick and accurate manner.
- Computer Aided Manufacturing (CAM) software is a special software that is used to generate programs for the CNC engraving machine.

3.1 Preparation of wooden wheel using computerized wood carving machine as shown in the Figure 3.1

- Preparation of CAD .dwg file
- Importing the file into wood carving machine for generating the profile using CAM software.

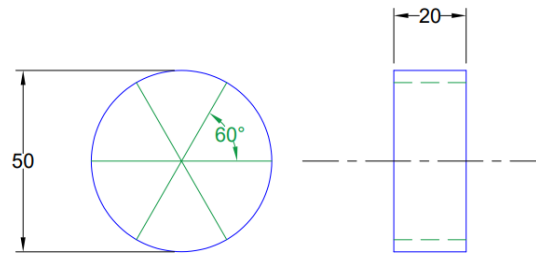


Figure 3.1 Wooden wheel

Try

1. Preparation of IARE lettering using CNC wood carving as shown in the Figure 3.2.



Figure 3.2 IARE lettering

Hint: Set the appropriate parameters to perform operation on the medium density fibreboard (MDF) wood material.

4. Introduction to pipe fitting and threading

Threaded fittings are used in non-critical applications and when service fluid is at ambient temperatures such as instrument air, plant air, cooling water, potable water etc. As they do not require welding, they are used at places where welding is not permitted.

- i) Preparation of Polyvinyl Chloride (PVC) material for pipe fitting
- ii) Making threads on PVC pipe using thread die sets
- iii) Fitting the threaded PVC pipe for T-shape using Tee joint

4.1 Preparation of PVC material for pipe threading and fitting as shown in the figure 4.1

- i) Start by cutting the two pipes to the required length, making sure that they are the same size and the ends line up properly.
- ii) Make the threads using die sets to one end of the pipes, then place it on top of the pipe where you want to join.
- iii) Then start rotating to get bonded together.

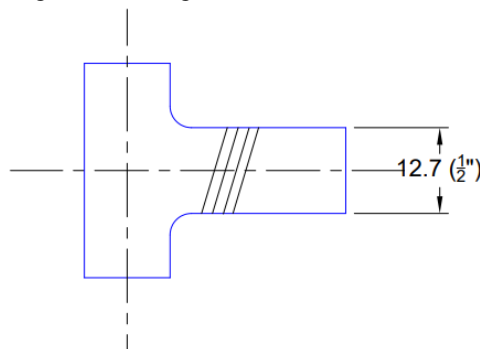


Figure 4.1 T Joint

Try

1. Preparation of galvanized steel I joint as shown in the Figure 4.2.
 - i) Start by cutting the two pipes to the required length, making sure that they are the same size and the ends line up properly.

- ii) Make the threads using die sets to one end of the pipes, then place it on top of the pipe where you want to join.
- iii) Then start rotating to get bonded together.

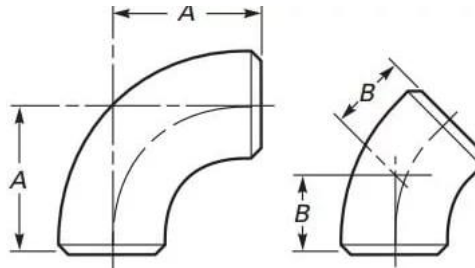


Figure 4.2. GI Elbow Joint

Dimensions:

A = 40mm

B = 30mm

Hint: Set the appropriate die sets in order to get perfect threading in pipe fittings.

5. Introduction to computer numerical control (CNC) lathe machines

A lathe is a machine tool that rotates a workpiece about an axis of rotation to perform various operations such as cutting, sanding, knurling, drilling, deformation, facing, and turning, with tools that are applied to the workpiece to create an object with symmetry about that axis.

- i) Operate with Computer Numerical Control (CNC) systems and provided with precise design instructions.
- ii) CNC Lathes are machine tools where the material or part is clamped and rotated by the main spindle, while the cutting tool that work on the material, is mounted and moved in various axis.

5.1 Preparation of Mild Steel (MS) material for step turning with grooving operation as shown in the Figure 5.1

- Inspect the mild steel raw material using Vernier calipers. The work piece is held in the chuck by placing it properly and tightening it using the chuck key.
- Now single point cutting tool is placed in the tool post and properly arranged to the centre of the work piece.
- Work piece is rotated by switching on the motor.
- Perform the facing operations on both sides and maintain the given dimensions.
- First the plain turning operation is carried out by placing the tool at 1 mm feed to the lathe axis.
- After that step turning operation is performed till the desired diameter is obtained.

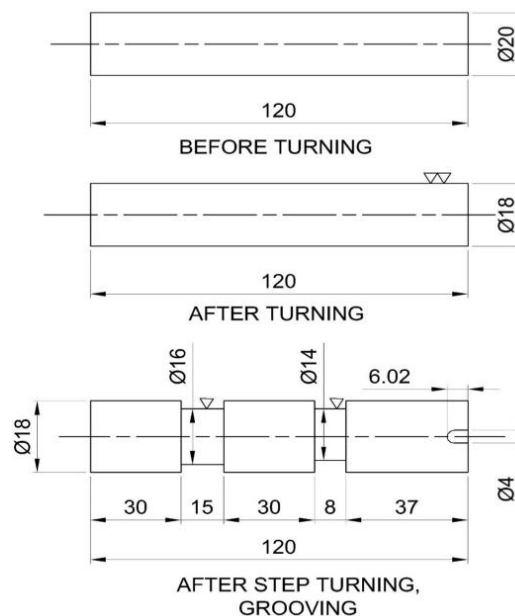


Figure 5.1. Step turning with grooving operation

Try

1. Preparation of Mild Steel (MS) material for step turning with taper operation as shown in the Figure 5.2.

- Select a tool bit to the desired size and shape of the groove required.
- Lay out the location of the groove.
- Set the lathe to half the speed for turning.
- Mount the workpiece in the lathe and set the tool bit to centre height.

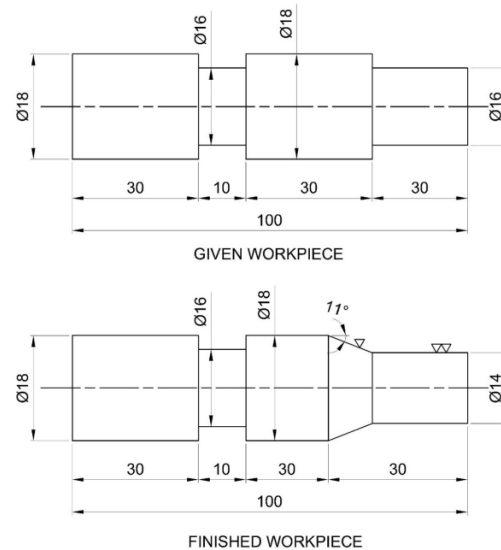


Figure 5.2. Step turning with taper operation

6. Introduction to conventional lathe machines

The conventional lathe machine is a standard lathe that is used for holding and turning various types of materials such as metal, wood, plastic etc. against a cutting tool in order to produce a cylindrical object. Besides this it can even perform many other functions like grinding, boring, threading, polishing, reaming, drilling etc.

- Operate with conventional lathe provides with manual design instructions.
- Lathes are machine tools where the material or part is clamped and rotated by the main spindle, while the cutting tool that work on the material, is mounted and moved in various axis.

6.1 Preparation of mild steel (MS) material for thread cutting and knurling operation as shown in the Figure 6.1

- Fix the job on the machine by using chuck key. Turn the job to the required diameter by fixing the single point cutting tool.
- Chamfer the edge and make an under cut at the other end.
- Engage the bed screw and perform the threading operation.
- Stop when the pitch is measured by the pitch gauge.
- Reverse the job and hold it carefully so that the threads are not damaged. Disengage the back gear and lead screw
- Hold the knurling tool against the rotating job.

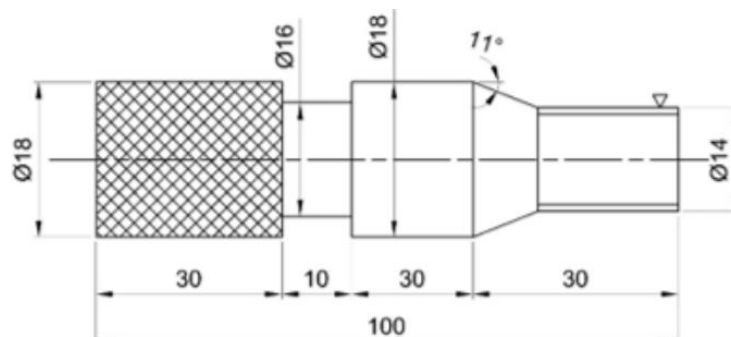


Figure 6.1. Thread cutting and knurling operation

Try

- Preparation of aluminum material for step turning with taper operation as shown in the Figure 6.2.

- Select a tool bit to the desired size and shape of the groove required.
- Lay out the location of the groove.
- Set the lathe to half the speed for turning.
- Mount the workpiece in the lathe and set the tool bit to centre height.

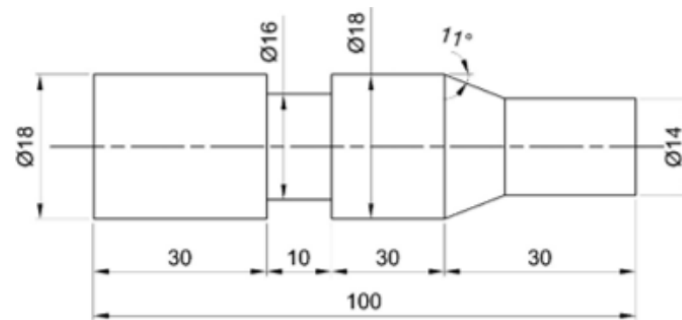


Figure 6.2. Step turning with taper operation

7. Introduction to milling machines

Milling is the process of machining using rotary cutters to remove material by advancing a cutter into a workpiece. This may be done by varying directions on one or several axes, cutter head speed, and pressure. Milling covers a wide variety of different operations and machines, on scales from small individual parts to large, heavy-duty gang milling operations. It is one of the most used processes for machining custom parts to precise tolerances.

- Milling machine employed in the metal removing operation in which the work is rigidly clamped on the table of the machine and the revolving cutter which has multiple teeth is mounted on the arbor.
- The cutter revolves at high speed and the work is fed slowly past the cutter.
- The work can be fed in a vertical, longitudinal, or cross direction depending upon the type of milling machine being used.
- As the work proceeds, the cutter-teeth removes the metal from the surface of the job(workpiece) to produce the desired shape.

7.1 Preparation of slotting operation as shown in Figure 7.1

- Keep the work piece on the working table in req. position with the help of holding device.
- Keep the cutting tool in the spindle and move the working table upward to give touch the surface of the work piece.
- Then give the power supply and move the work table forward and backward with the help of lever.
- Repeat the same procedure by changing the feed rate in upward and cross direction to get the req. dimension of slot on the work piece.

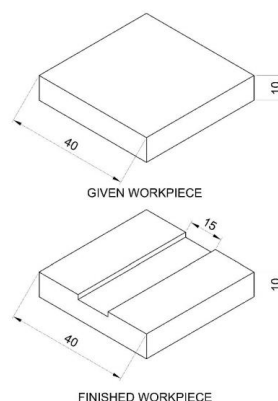


Figure 7.1 Slotting operation

Try

1. Perform the boring and reaming operation on a rectangular work piece to obtain the required dimensions using vertical milling machine as shown in Figure 7.2.

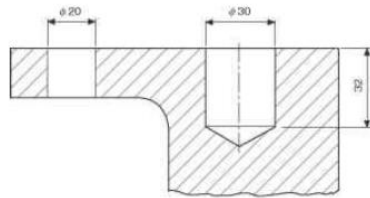


Figure 7.2. Boring and reaming operation

8. Introduction to shaping machines

A shaping machine is a mechanical device used to shape and form metal workpieces. It operates by removing material through a reciprocating cutting motion, resulting in the desired shape or contour. Shaping machines are commonly used in metalworking industries for various applications, including creating flat surfaces, slots, and grooves.

- A rigid table on the machine supports the workpiece. Over the workpiece, the ram moves back and forth as shown in the animation above. A vertical tool slide is adjusted to either side of the vertical plane along the stroke axis, which is located at the front of the ram.
- The geometry of the linkage causes the ram to travel more quickly on the return stroke than the forward stroke (cutting stroke). As the shaper works on the quick return mechanism, the sliding action of the slider is aided by the rotating link.
- One of the four mechanisms i.e. crank and slotted, whitworth quick return, hydraulic, and automatic table feed mechanism, is responsible for the quick return mechanism and reciprocating movement of the ram. The automatic table feed is commonly used today which employs a pawl and ratchet mechanism in a shaping machine.

8.1 Preparation of V-groove operation as shown in Figure 8.1

- The job is fixed on a vice and the tool is fixed on tool post.
- The stroke of ram is adjusted to required length and machine is switched on.
- Always during machining, the job should be properly fixed with the half of try Square and vice to get a right-angle surface after machining.
- After completion of work, the job should be filled help of file before fixing the job, V block dimensions are marked on the job with the help of dot punch.
- The tool head should be rotated at 45° to make the V- groove.
- The feed is given such that the tool moves gradually on either side of the middle line.

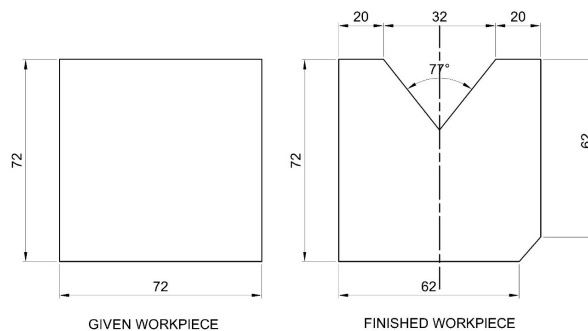


Figure 8.1 V-groove operation

Try

- Perform the key ways on a cylindrical work piece to obtain the required dimensions using shaping machine as shown in Figure 8.2

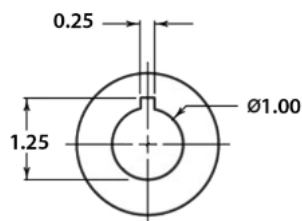


Figure 8.2 Key ways on a cylindrical work piece

9. Introduction to electrical wiring practices for domestic appliances

The Electrical Wiring Systems are mostly standardized with several rules, regulations, and laws. Electrical Wiring must be installed correctly and safely in accordance with electrical regulations and standards. If the electrical wiring is carried out incorrectly or without confirming to any standard, then it may lead to incidents like short circuits, electric shocks, damage the device / appliance or leads to the malfunctioning of device which further causes for the reduction of device life.

- i) Before starting any installation work, the first and foremost thing is the concern of safety of the personnel. Electricity is dangerous and direct or indirect contact of electrical equipment or wires with the power turned ON can result in serious injuries or sometimes even causes death. Follow the below steps to maintain the safety at the workplace.
- ii) Several factors must be considered before the actual installation work to be done for residential, commercial, or industrial wiring. These factors include type of building construction, type of ceiling, wall, and floor construction, wiring methods, installation requirements, etc.

9.1 Preparation of wiring for a stair case arrangement using a two-way switch as shown in Figure 9.1

- Mark switch and bulb location points and draw lines for wiring on the wooden Board.
- Place wires along the lines and fix them with the help of clips.
- Fix the two-way switches and bulb holder in the marked position on the wooden Board.
- Complete the wiring as per the wiring diagram.
- Test the working of the bulbs by giving electric supply to the circuit

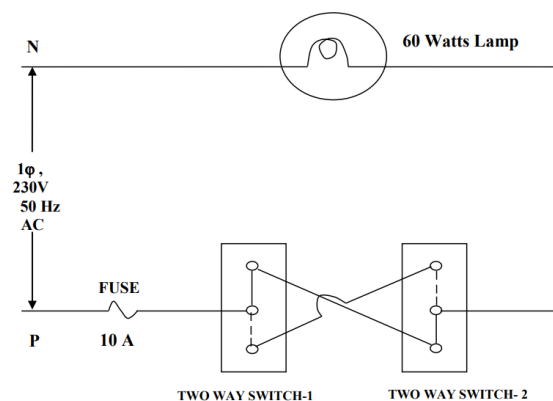


Figure 9.1 Circuit diagram-staircase wiring

Try

1. Prepare wiring for a tube light with switch control as shown in Figure 9.2.
 - Mark the switch and tube light location points and draw lines for wiring on the wooden board.
 - Place wires along the lines and fix them with the help of clips.
 - Fix the switch and tube light fitting in the marked positions.
 - Complete the wiring as per the wiring diagram.
 - Test the working of the tube light by giving electric supply to the Circuit.

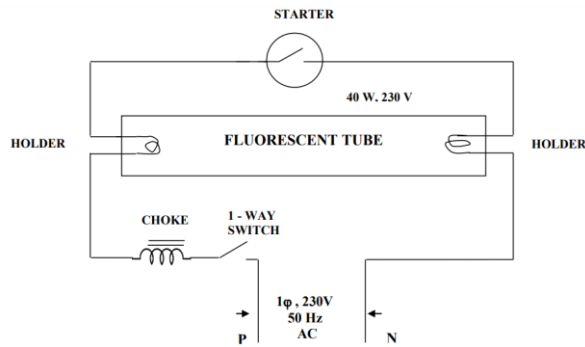


Figure 9.2 Circuit diagram-tube light

10. Introduction to soldering and desoldering practice

Soldering is defined as "the joining of metals by a fusion of alloys, which have relatively low melting points". In other words, you use a metal that has a low melting point to adhere the surfaces to be soldered together. Soldering is more like gluing with molten metal than anything else. Soldering is also a must have skill for all sorts of electrical and electronics work. It is also a skill that can only be developed with practice.

- i) Soldering requires two main things: a soldering iron and solder. Soldering irons are the heat source used to melt solder. Irons of the 15W to 30W range are good for most electronics/printed circuit board work.
- ii) Using anything higher in wattage and you risk damaging either the component or the board. Note that you should not use so-called soldering guns. These are very high wattage and generate most of their heat by passing an electrical current through a wire. Because of this, the wire carries a stray voltage that could damage circuits and components. The choice of solder is also important.
- iii) One of the things to remember is to never use acid core solder. Acid core solder will corrode component leads, board traces and form conductive paths between components.

10.1 Preparation of soldering from a circuit board as shown in Figure 10.1

- All parts must be clean and free from dirt and grease.
- Try to secure the work firmly.
- "Tin" the iron tip with a small amount of solder. Do this immediately, with new tips being used for the first time.
- Clean the tip of the hot soldering iron on a damp sponge.
- Many people then add a tiny amount of fresh solder to the cleansed tip.
- Heat all parts of the joint with the iron for under a second or so.
- Continue heating, then apply sufficient solder only, to form an adequate joint.
- Remove and return the iron safely to its stand.
- It only takes two or three seconds at most, to solder the average printed circuit board (PCB). joint.
- Do not move parts until the solder has cooled.

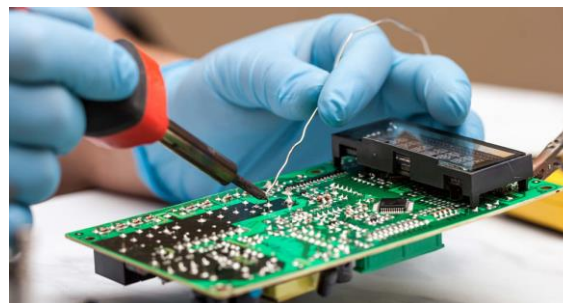


Figure 10.1 Soldering operation

Try

1. Perform desoldering operation from a circuit board as shown in Figure 10.2.
 - Heat up the solder with the iron.
 - Slide the iron up the pins to bring most of the solder away from the joint.

- Using pliers, gently pull at the components to remove their pins from the pin holes while they are still hot. It's a good idea to pull by their leads as opposed to on the components themselves to maintain the quality of the component.

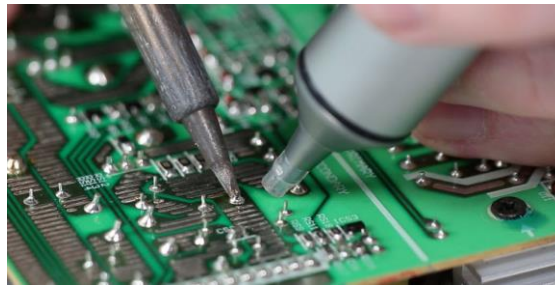


Figure 10.2 De-Soldering operation

11. Introduction to troubleshooting the ceiling fan and mixer grinder

In generally most ceiling fan and table fan have a capacitor start permanent capacitor motors which a difference from usual motor that it central position (Rotor/ Armature) remains fixed, while the outer portion rotates blades are mounted on the outer shaft when the motor is energized the blade cause to rotate and to circulate the surroundings are depends on the speed of fan. A regulator is connected in series with fan at different tapings hence the speed of the fan consist of a number of parts are connected together as a shaft to avoid loose fittings the parts are located bolts, split, pins and bearing lock.

- Usually, a ceiling fan and table fan consist of a capacitor start and run motor. Capacitor are always connected in the circuit the advantages of leaving the capacitor permanently in circuit.
- It has one starting winding in series with one capacitor and running winding since the capacitor remains in the circuit permanently. This motor is often referred to as permanent. Split capacitor runs motor and behaves practically like an unbalanced 2phase motor.

11.1 Perform the maintenance of ceiling fan and ending the trouble shoot problems as shown in Figure 11.1

- The rapid spinning and vibrations your fan's components are subjected to can cause them to work loose and wear out. Inspecting the fan every two or three months you use it helps keep the fan working efficiently and extends its lifespan.
- One of the most common problems is a loose mount, which can cause your fan to wobble. A wobbling fan is unlikely to fall, but it can cause the light fixture to fall, so it's not something to ignore.
- Blades that are misaligned or out of balance can also cause wobbling. Over time, one or more blades can become warped, bent or otherwise damaged. Even a minor difference can disrupt the fan's normal performance, so check each blade closely.

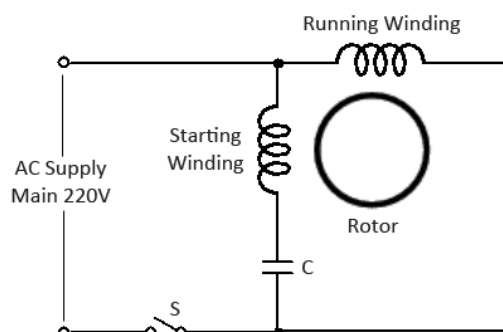


Figure 11.1 Ceiling fan circuit diagram

Try;

1. Perform the maintenance for mixer grinder from a circuit board as shown in Figure 11.2.

- Universal Series Motor
- Three Position Speed Control Rotary Switch
- Thermal Overload Relay or overload switch
- Indicator Light
- Power Switch

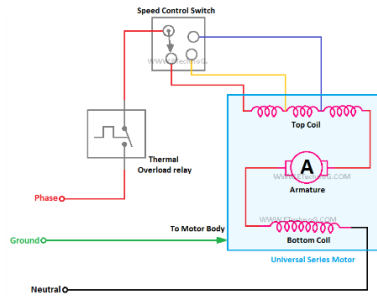


Figure 11.2. Mixer grinder circuit diagram

12. Introduction to 6 axis articulated robotic arm

ARISTO is six axis articulated robotic arm of industrial for training and research and is manufactured to industrial standards. The robot is capable of lifting up to 2.5kg of pay load. The robot can be used with pneumatic or electrical grippers. ARISTO has simulation software that allows the user to learn robot functions, application & programming.

- The evolution of the performance of robots and programming software provides new machining solutions. For complex parts, six axes robots offer more accessibility than a machining center CNC 5 axis and allow the integration of additional axes to extend the workspace.
- Robots have seen in recent years an expansion of their field of use with new requirements related to the increasing use of composites. The robots are then considered for machining operations (polishing, cutting, drilling etc.) that require high performance in terms of position, orientation, followed by trajectory precision and stiffness.

12.1 Preparation of articulated robot for lifting load as shown in Figure 12.1

- A load lifting robot is a type of industrial robot that is used for handling and placing products on a production line.
- They are typically used in high-volume manufacturing settings, where they can quickly and accurately place products onto conveyor belts or other production equipment

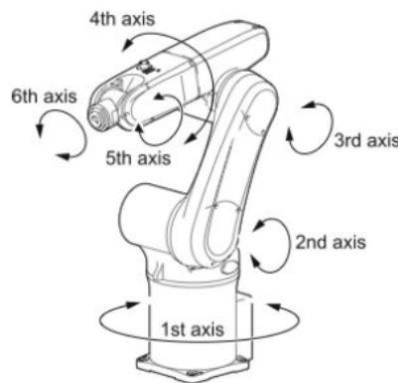


Figure 12.1 Six Axis Aristo Robot

Try

1. Perform the pick and place operation for the articulated robot as shown in Figure 12.2

- A pick-and-place robot is a type of industrial robot that is used for handling and placing products on a production line.
- They are typically used in high-volume manufacturing settings, where they can quickly and accurately place products onto conveyor belts or other production equipment.



Figure 12.2. Pick and place operation

13. Introduction to FANUC simulator

The FANUC CNC Simulator brings the world's most popular control right into your workplace training room, providing hands-on training for FANUC CNC operation without the need for a full machine. Add Machining Simulation Software to the CNC Simulator for advanced machine simulation capability.

- This PC-based platform is perfect for training and designing part programming. The CNC Machining Simulation software provides a digital twin of the machine tool producing the real-world cutting process. This provides you with the most realistic simulation of the actual machining on your floor. To prepare the industry for more complex machining knowhow, a 5-axis machining training option is now available.
- For companies needing a more tailored training offering, FANUC America's CNC Hardware Simulators are fully functional CNCs that include the panel and operating system. More robust than our CNC Simulators, our CNC Hardware Simulators are complete control simulators customized to address your specific workplace training needs. Operations needing a particular CNC model for their shopfloor training will benefit from these simulators.

13.1 Preparation of milling and lathe system switchable on one simulator as shown in Figure 13.1

- Switchable mill and lathe (turning) system in one simulator
- 3-axis milling / 2-axis turning system plus one spindle
- Manual Guide is installed for conversational program creation and 3D simulation
- Imperial / metric switchable
- 512KB part-program storage, with 400 registered programs
- 32 tool offset pairs
- Workpiece coordinates G52 – G59 plus 48 additional on mill



Figure 13.1 FANUC Simulator

Try

1. Perform the combination of CNC Simulator with CNC machining simulation software as shown in Figure 13.2.
 - The CNC Machining Simulator carries all the same features as the standard simulator with the addition of an internal PC, which will provide a virtual mill or lathe with real kinematics, allowing the user to view the live machine movement, tooling and part machining.
 - Tool and workpiece setup is required and colored back plot, collision detection and sound bring the machine operating experience to life.



Figure 13.2 CNC Simulator with CNC machining simulation

14. Demonstration to cylindrical grinding machine

Most commonly, cylindrical grinding is used for grinding pieces with a central axis of rotation, like rods and cylinders. This process involves using a cylindrical grinder, which is a type of machinery categorized by rotation style and wheel device.

- A grinding machine uses an abrasive product usually a rotating wheel to shape and finish a workpiece by removing metal and generating a surface within a given tolerance. A grinding wheel is made with abrasive grains bonded together. Each grain acts as a cutting tool, removing tiny chips from the workpiece.

14.1 Demonstration on industry standard grinding as shown in Figure 14.1

- The ECO-200 Micromatic industry standard universal cylindrical grinding machine is used for grinding of components up to an accuracy of 5m and used for projects.



Figure 14.1 Cylindrical griding

Try

1. Demonstration grinding methods and machines
 - Grinding, or abrasive machining, once performed on conventional milling machines, lathes and shapers, is now performed on various types of grinding machines.
 - Grinding machines have advanced in design, construction, rigidity and application far more in the last decade than any other standard machine tool in the manufacturing industry. Grinding machines fall into five categories: surface grinders, cylindrical grinders, centerless grinders, internal grinders and specials.

V. TEXT BOOKS:

1. Hajra Choudhury S.K., Hajra Choudhury A.K. and NirjharRoy S.K., *Elements of Workshop Technology*, Media promoters and publishers private limited, Mumbai, 2020.
2. Kalpakjian S, Steven S. Schmid, *Manufacturing Engineering and Technology*, Pearson Education India Edition, 7th Edition, 2019.

VI. REFERENCE BOOKS:

1. Gowri P. Hariharan, A. Suresh Babu, *Manufacturing Technology – I*, Pearson Education, 2018.
2. Roy A. Lindberg, *Processes and Materials of Manufacture*, Prentice Hall India, 4th Edition, 2017.
3. P.N., *Manufacturing Technology*, Vol. I and Vol. II, Tata McGraw-Hill House, 2017.
4. Rupinder Singh, J. Paulo Davim, *Additive Manufacturing: Applications and Innovations*, CRC Press, 2nd Edition, August, 2021.
5. Jeyaprakash Natarajan, Muralimohan Cheepu, Che-Hua Yang, *Advances in Additive Manufacturing Processes*, Bentham Books, 4th Edition, September, 2021.

VII. ELECTRONIC RESOURCES:

1. <https://elearn.nptel.ac.in/shop/iit-workshops/ongoing/additive-manufacturing-technologies-for-practicing-engineers/>.
2. https://akanksha.iare.ac.in/index?route=course/details&course_id=94.

VIII. MATERIALS ONLINE:

1. Course Template
2. Lab manual

COURSE CONTENT

ELECTRICAL CIRCUITS LABORATORY								
I Semester: EEE / ECE								
Course Code	Category	Hours / Week			Credits	Maximum Marks		
AEED04	Foundation	L	T	P	C	CIA	SEE	Total
		-	-	2	1	40	60	100
Contact Classes: Nil	Tutorial Classes: Nil	Practical Classes: 45			Total Classes: 45			
Prerequisite: There are no prerequisites to take this course								

I. COURSE OVERVIEW:

The course is designed to provide students with hands-on experience and practical skills in the field of electrical engineering. The course serves as a complement to theoretical concepts learned in the concurrent electrical circuits course. Through a series of structured experiments, students gain an in-depth understanding of fundamental electrical circuits, components, measurement techniques, and troubleshooting procedures.

II. COURSES OBJECTIVES:

The students will try to learn

1. The gap between theoretical knowledge and practical applications by exposing students to a wide range of electrical components and circuit configurations.
2. The essential skills in circuit design, measurement, testing, and analysis using laboratory equipment such as multimeters, and power supplies.
3. The basic laws, network reduction techniques and theorems for different circuits.

III. COURSE OUTCOMES:

At the end of the course students should be able to:

- CO 1: Solve the source resistance, currents, voltage and power using various laws associated with electrical circuits
- CO 2: Analyze the alternating quantities for different periodic waveforms.
- CO 3: Perform the superposition principle, reciprocity and maximum power transfer condition for the electrical network with DC excitation.
- CO 4: Demonstrate Thevenin's and Norton's theorems to reduce complex networks into simple equivalent networks with DC excitation.
- CO 5: Apply Faraday's laws of electromagnetic induction for calculating the various performance parameters in magnetic circuits.
- CO 6: Make use of two port network parameters for enumerating the symmetry, reciprocity, internal voltages and currents in the various electrical circuits

DO's

1. Once the operation is completed pull the plug itself rather chord attached to it.
2. To repair the equipment switch-off the supply and go on.
3. To operate the equipment on supply, see that hands are dry, if that is not possibly hide the hand in the pockets.
4. If a person comes in contact with current unexpectedly don't touch the person with hands but immediately use any insulator material and shut down the power (like leather belts, wood and plastic bars etc).
5. If water is nozzles on the equipment, immediately shunt down the power using circuit breaker or pull out the plug.
6. Use the connecting wires of good continuity, short circuit of connecting wire leads damage of circuit parameters.

DON'Ts

1. Do not wear loose clothing and do not hold any conducting materials in contact with skin when the power is on.
2. Do not pull out the connections until unless all the currents are dead.
3. Do not wait for switches to de-magnetize when there is a delay but pull out the plug.
4. Do not overload the circuit by plugging in too many appliances.
5. If you are mentally and physically stressed don't operate the power equipment.
6. Never operate the equipment under wet conditions.
7. Do not interconnect two or more wires, take appropriate length of wire.

SAFETY NORMS

1. The lab must be equipped with fire extinguisher.
2. See that the connections are made tight.
3. Use single plug for each equipment.
4. Cover the body completely to avoid arc effect.
5. To change the connections during the experiment, switch off the supply and carry on.
6. Used equipment may get heated, so take care handling the equipment after it is used.
7. Do the wiring, all set ups and check the circuit connections before the supply is on

IV. COURSE CONTENT:

EXERCISES FOR ELECTRICAL CIRCUITS LABORATORY

Note: Students are encouraged to bring their own laptops for laboratory practice session

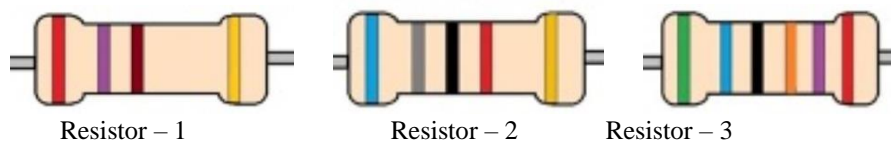
1. Getting Started Exercises

1.1 Introduction to electrical circuits

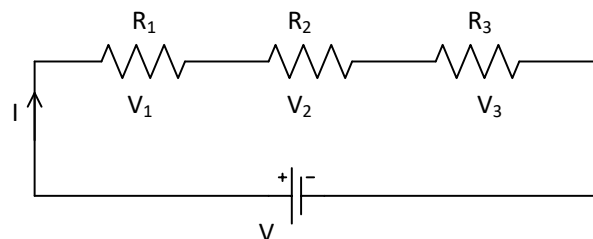
1. Understand the basic electrical equipments used in the laboratory.
2. Become familiar with the operation and usage of basic DC electrical laboratory devices, namely DC power supplies and digital multimeters.
3. Learn the measurement of resistance values using colour code and digital multimeter.
4. Learn the basics of circuit design using Simulink.

Try

1. Calculate the resistance value of Resistor – 1, Resistor – 2 and Resistor – 3 using colour code and verify using a digital multimeter

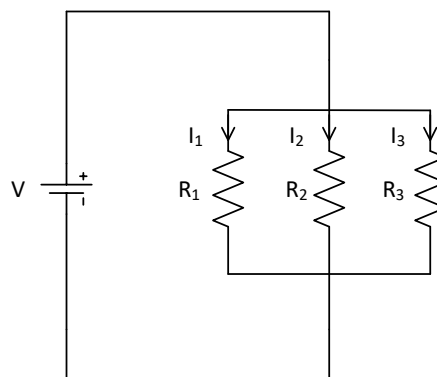


2. Design Circuit – 1 using Simulink and find the voltages V_1 , V_2 , V_3 and Current I . Where $V_s = 6\text{ V}$, $R_1 = 100\ \Omega$, $R_2 = 220\ \Omega$, $R_3 = 1\text{ k}\Omega$



Circuit – 1

3. Design Circuit – 2 using Simulink and find the currents I_1 , I_2 and I_3 . Where $V_s = 6\text{ V}$, $R_1 = 100\ \Omega$, $R_2 = 220\ \Omega$, $R_3 = 1\text{ k}\Omega$

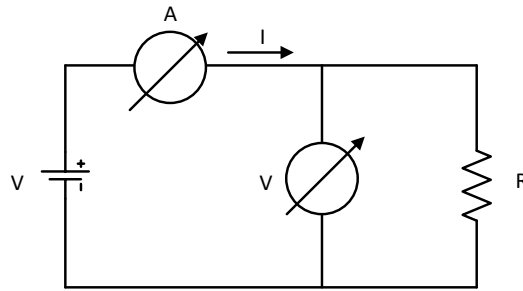


Circuit – 2

2. Exercises on Basic Electrical Circuit Law's

2.1 Ohm's law

Examine Ohm's law of Circuit – 3 and draw the V-I characteristic of linear resistors $R = 1\text{ k}\Omega$.



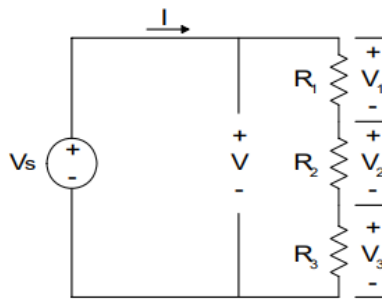
Circuit - 3

Try

1. Verify Ohm's law of Circuit – 3 using Simulink and draw the V-I characteristics of a linear resistor $R = 470 \text{ k } \Omega$.
2. An electric heater takes 1.48 kW from a voltage source of 220 V . Find the resistance of the heater.

2.2 Kirchhoff's voltage law

Examine Kirchhoff's voltage law using basic series DC Circuit - 4 with resistors. Where $V_s = 6 \text{ V}$, $R_1 = 100 \text{ } \Omega$, $R_2 = 220 \text{ } \Omega$, $R_3 = 1 \text{ k } \Omega$.



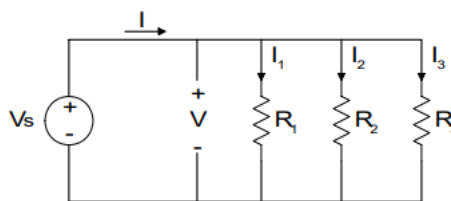
Circuit – 4

Try

1. Design and Verify Kirchhoff's voltage law for Circuit – 4 using Simulink.
2. Determine the voltage V_2 by replacing the resistor $R_2 = 150 \text{ } \Omega$ in Circuit – 4.
3. Find the total current I flowing through the Circuit – 4.

2.3 Kirchhoff's current law

Examine Kirchhoff's current law using basic parallel DC Circuits - 5 with resistors. Where $V_s = 6 \text{ V}$, $R_1 = 100 \text{ } \Omega$, $R_2 = 220 \text{ } \Omega$, $R_3 = 1 \text{ k } \Omega$



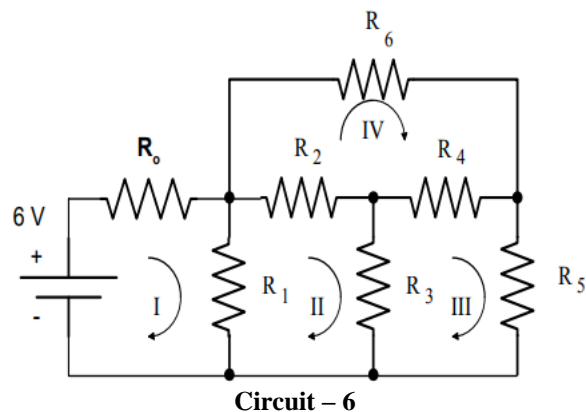
Circuit – 5

Try

1. Design and Verify Kirchhoff's current law for Circuit – 5 using Simulink.
2. Determine the current I_2 by replacing the resistor $R_2 = 150 \text{ } \Omega$ in Circuit – 5.

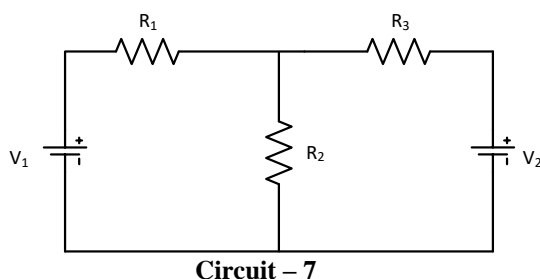
3. Exercises on Mesh Analysis

Determine mesh currents in the complex electrical Circuit - 6 by using principles of basic electrical circuits. Where $R_0 = 47 \text{ } \Omega$, $R_1 = 100 \text{ } \Omega$, $R_2 = 220 \text{ } \Omega$, $R_3 = 1 \text{ k } \Omega$, $R_4 = 150 \text{ } \Omega$, $R_5 = 82 \text{ } \Omega$, $R_6 = 100 \text{ } \Omega$.



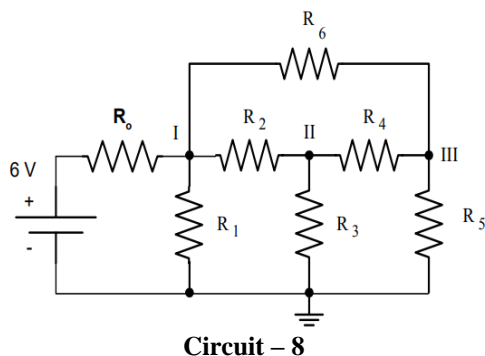
Try

1. Use Simulink to simulate the Circuit – 6 for determining the current flowing through each resistor and compare these values to those you obtained from your experiments.
2. Find the current flowing through the resistor R_2 and R_3 by replacing the values of the resistors $R_2 = 100\ \Omega$, $R_3 = 470\ \Omega$ in Circuit – 6.
3. Find the current flowing through the resistor R_3 for Circuit – 7 using mesh analysis when $V_1 = 10\text{ V}$, $V_2 = 6\text{ V}$, $R_1 = 100\ \Omega$, $R_2 = 220\ \Omega$ and $R_3 = 1\text{ k}\ \Omega$.



4. Exercises on Nodal Analysis

Determine nodal voltages in complex electrical Circuit – 8 by using principles of basic electrical circuits. Where $R_0 = 47\ \Omega$, $R_1 = 100\ \Omega$, $R_2 = 220\ \Omega$, $R_3 = 1\text{ k}\ \Omega$, $R_4 = 150\ \Omega$, $R_5 = 82\ \Omega$, $R_6 = 100\ \Omega$.



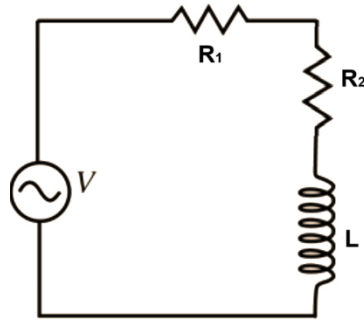
Try

1. Use Simulink to simulate the Circuit – 8 for determining the voltage across each resistor and compare these values to those you obtained from your experiments.
2. Determine the voltage at node - II by replacing the resistor $R_2 = 150\ \Omega$ in Circuit – 8.
3. Find the node voltage for Circuit – 7 using nodal analysis when $V_1 = 10\text{ V}$, $V_2 = 6\text{ V}$, $R_1 = 100\ \Omega$, $R_2 = 220\ \Omega$ and $R_3 = 1\text{ k}\ \Omega$.

5. Exercises on Characteristics of Periodic Waveforms

5.1 Sinusoidal waveform

Calculate the average value, RMS value, form factor and peak factor of the sinusoidal waveform for Circuit – 9. Where $V = 5\text{ V}$, $R_1 = 1\text{ k}\ \Omega$, $R_2 = 100\ \Omega$ and $L = 10\text{ mH}$.



Circuit – 9

Try

1. Use Simulink to simulate the Circuit – 9 for determining the average value, RMS value, form factor and peak factor of the sinusoidal waveform.
2. Find the Average Value, RMS Value, form Factor and Peak Factor for the wave shown in the Figure 1.

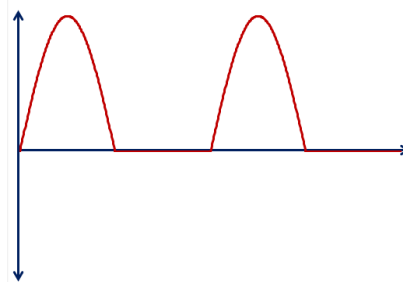


Figure 1. Periodic wave

5.2 Square waveform

Calculate the average value, RMS value, form factor and peak factor of the square waveform shown in Figure 2.

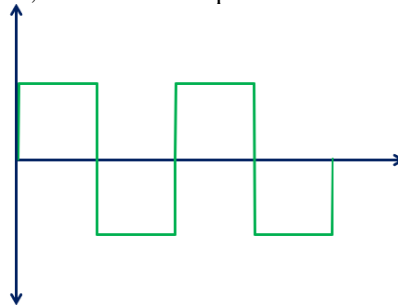


Figure 2. Square wave

Try

1. Find the Average Value, RMS Value, form Factor and Peak Factor for the wave shown in Figure 3.

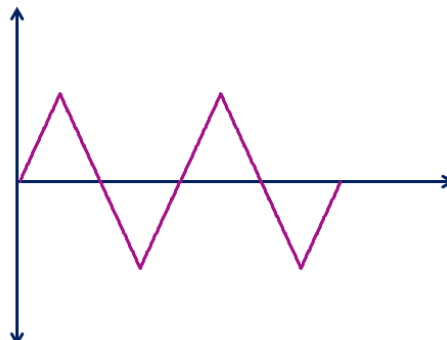


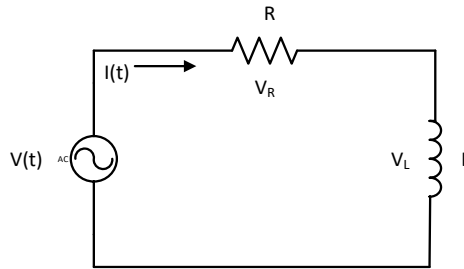
Figure 3. Triangular wave

2. Sketch both sinusoidal and square waveforms on the same set of axes over two complete cycles and label the key points.

6. Exercises on Determination of Circuit Impedance

6.1 Impedance of series RL circuit

Determine the impedance of the series RL circuit shown in Circuit – 10 by considering $V(t) = 5\text{ V}$, $R = 150\ \Omega$ and $L = 10\text{ mH}$.



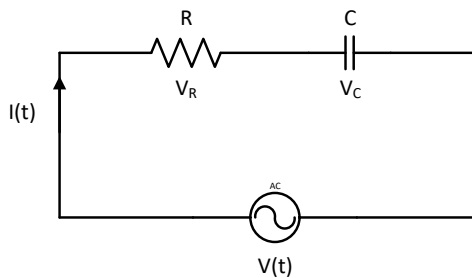
Circuit – 10

Try

1. Use Simulink to simulate the RL circuit response for Circuit – 10 and compare these values to those you obtained from your experiments.
2. Calculating the angular frequency (ω) of the AC source.
3. Sketch the phasor diagram for the RL circuit.

6.2 Impedance of series RC circuit

Determine the impedance of the series RC circuit shown in Circuit – 11 by considering $V(t) = 5\text{ V}$, $R = 150\ \Omega$ and $C = 0.1\ \mu\text{F}$.



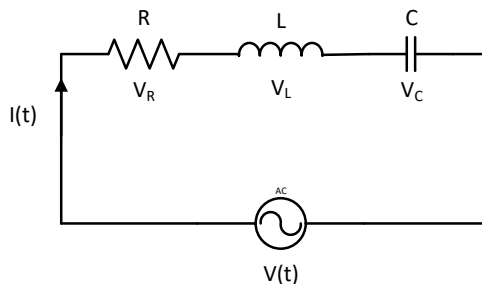
Circuit – 11

Try

1. Use Simulink to simulate the RC circuit response for Circuit – 11. Compare these values to those you obtained from your experiments.
2. Calculating the angular frequency (ω) of the AC source.
3. Sketch the phasor diagram for the RC circuit.

6.3 Impedance of series RLC circuit

Determine the impedance of the series RLC circuit shown in Circuit – 12 by considering $V(t) = 5\text{ V}$, $R = 150\ \Omega$, $L = 10\text{ mH}$ and $C = 0.1\ \mu\text{F}$.



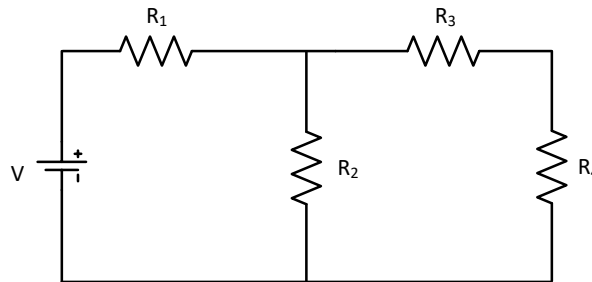
Circuit – 12

Try

1. Use Simulink to simulate the RLC circuit response for Circuit – 12 and compare these values to those you obtained from your experiments.
2. Calculating the angular frequency (ω) of the AC source.
3. For the RLC circuit, calculate the phase angle (ϕ) between the current and voltage.
4. Sketch the phasor diagram for the RLC circuit, representing the impedance, current, and voltage phasors.

7. Exercises on Thevenin's Theorem

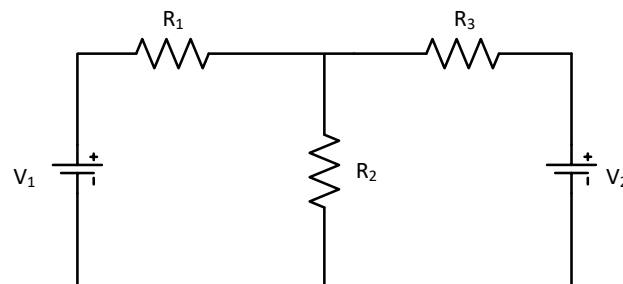
1. Determine Thevenin's equivalent voltage (V_{th}) and resistance (R_{th}) at R_4 by applying Thevenin's theorem for Circuit – 13.
2. Determine load or unknown current through a R_4 resistor using Thevenin's equivalent circuit. Where $V = 10V$, $R_1 = 100\ \Omega$, $R_2 = 220\ \Omega$, $R_3 = 1k\ \Omega$ and $R_4 = 150\ \Omega$.



Circuit – 13

Try

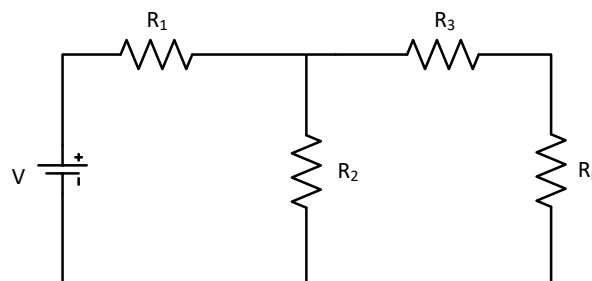
1. Use Simulink to simulate the Circuit – 13 for determining the current through R_4 resistor using Thevenin's theorem and compare this value to those you obtained from your experiment.
2. Find the current through R_4 resistor using any circuit reduction technique and verify this value to those you obtained from Thevenin's theorem.
3. Find the current flowing through R_2 resistor in Circuit – 14 using Thevenin's theorem for the below circuit. Where $V_1 = 10\ V$, $V_2 = 5\ V$, $R_1 = 100\ \Omega$, $R_2 = 220\ \Omega$, $R_3 = 1k\ \Omega$.



Circuit – 14

8. Exercises on Norton's Theorem

1. Find Norton equivalent current (I_N) and resistance (R_N) by considering $R_L = 150\ \Omega$ resistor for the Circuit – 15 by applying Norton's theorem.
2. Find load or unknown current through R_L resistor using Norton's equivalent circuit. Where $V = 10\ V$, $R_1 = 100\ \Omega$, $R_2 = 220\ \Omega$ and $R_3 = 1k\ \Omega$.



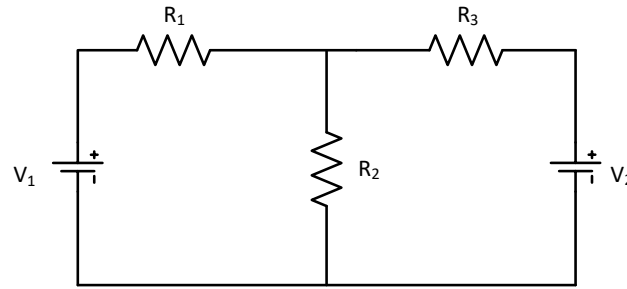
Circuit – 15

Try

1. Use Simulink to simulate the Circuit – 15 for determining the current through R_L resistor and compare this value to those you obtained from your experiment.
2. Find the current through R_L using any circuit reduction technique and verify this value to those you obtained from Norton's theorem.

9. Exercises on Superposition Theorem

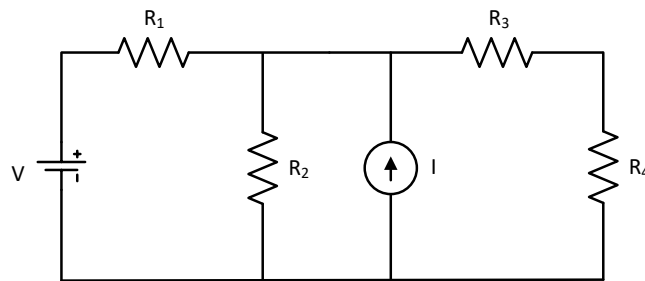
Investigate the current through R_2 resistor using superposition theorem to multiple DC source
Circuit – 16.



Circuit – 16

Try

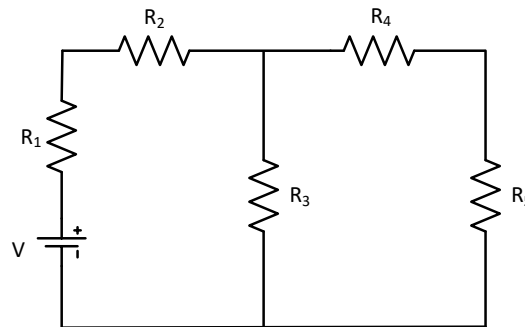
1. Use Simulink to simulate the Circuit – 16 for determining the current through a resistor R_2 and compare this value to those you obtained from your experiment.
2. Find the current through R_2 in Circuit – 16 using any circuit reduction technique and verify this value to those you obtained from the superposition theorem.
3. Find the current through the R_4 resistor using the superposition theorem for the Circuit – 17.



Circuit – 17

10. Exercises on Reciprocity Theorem

Understand the reciprocity theorem by analyzing Circuit – 18 with interconnected components using fundamental circuit laws where $V = 10\text{ V}$, $R_1 = 100\ \Omega$, $R_2 = 220\ \Omega$, $R_3 = 1\text{ k}\Omega$, $R_4 = 150\ \Omega$ and $R_5 = 82\ \Omega$.



Circuit – 18

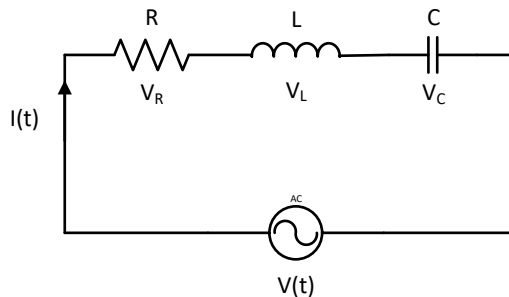
Try

1. Use Simulink to simulate Circuit–18 for determining the current through R_5 and R_1 by interchanging the voltage source in series with the R_5 resistor and compare this value to those you obtained from your experiment.
2. Find the current through R_3 using Thevenin's theorem.

11. Exercise on Series and Parallel Resonance

11.1 Series resonance

Study the frequency response characteristics of series resonance circuit of Circuit – 19 and hence to determine the inductance, bandwidth and quality factor of the circuit. Where $R = 1\text{ k}\Omega$, $L = 10\text{ mH}$, $C = 0.1\ \mu\text{F}$.



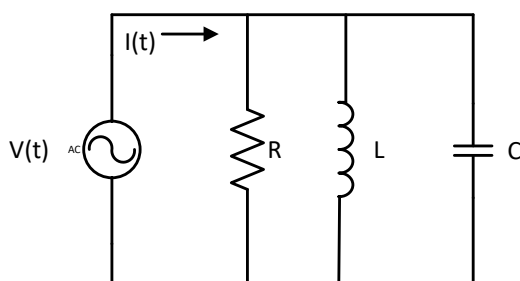
Circuit – 19

Try

1. Use Simulink to simulate Circuit–19 and plot the waveforms V_R , V_L and V_C versus frequency.
2. For series resonant circuit find I , V_R , V_L and V_C at resonance when $V = 10\text{ V}$, $R = 2\ \Omega$, $X_L = 10\ \Omega$ and $X_C = 10\ \Omega$.
3. If the resonant frequency is 5000 Hz, find bandwidth.

11.2 Parallel resonance

Study the frequency response characteristics of parallel resonance circuit of Circuit – 20 and hence to determine the inductance, bandwidth and quality factor of the circuit. Where $R = 1\text{ k}\Omega$, $L = 10\text{ mH}$, $C = 0.1\ \mu\text{F}$.



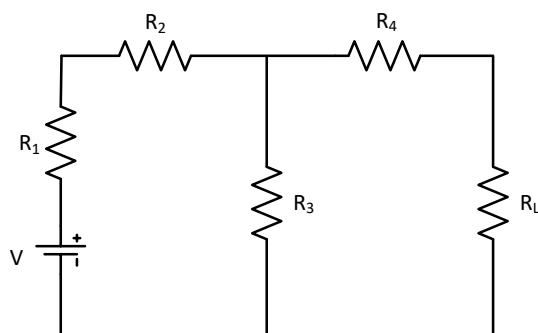
Circuit – 20

Try

1. Use Simulink to simulate parallel resonance circuit of Circuit – 20 and plot magnitude of response vs frequency. Considered the frequency range from 600 Hz to 2,500 Hz in increments of 100 Hz and compare with the experimental plot.

12. Exercise on Maximum Power Transfer Theorem

1. Find the value of R_L at which maximum power is transferred to the load in Circuit – 21.
2. Determine the maximum power transferred to the load. Where $V = 10\text{ V}$, $R_1 = 100\ \Omega$, $R_2 = 220\ \Omega$, $R_3 = 1\text{ k}\Omega$, $R_4 = 150\ \Omega$.



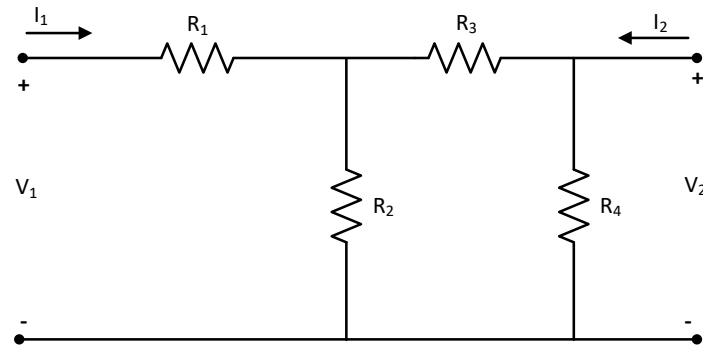
Circuit – 21

Try

1. If the internal resistance of a voltage source is adjustable, and the load resistor is fixed. Determine the range of source resistance values that will result in effective power transfer. Calculate the corresponding load power for each extreme of the source resistance range.
2. Use Simulink to simulate Circuit – 21 for determining the current through R_L and compare this value to those you obtained from your experiment.

13. Exercise on Z and Y Parameters

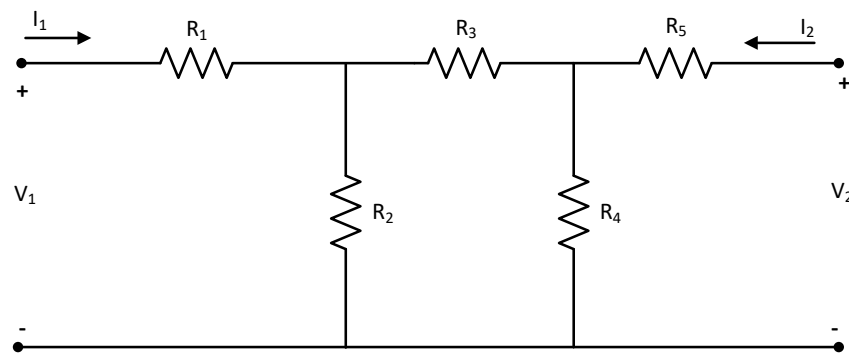
Calculate Z-parameters and Y-parameters for Circuit – 22, and analyze the network's response to different conditions (open and short circuits). Where $R_1 = 100\ \Omega$, $R_2 = 220\ \Omega$, $R_3 = 1\text{ k}\Omega$ and $R_4 = 150\ \Omega$.



Circuit – 22

Try

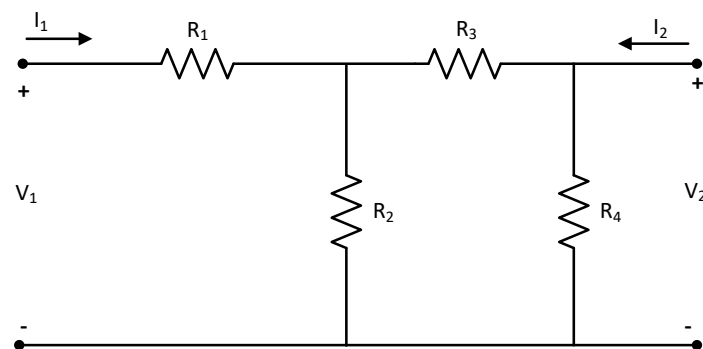
1. Use Simulink to simulate Circuit–22 for determining the Z and Y parameters and compare the values to those you obtained from your experiment.
2. Calculate Z parameters and convert in terms of Y parameters for Circuit – 23. Where $R_1 = 100\ \Omega$, $R_2 = 220\ \Omega$, $R_3 = 1\text{ k}\Omega$, $R_4 = 150\ \Omega$ and $R_5 = 82\ \Omega$.



Circuit – 23

14. Exercise on H and ABCD Parameters

Calculate H and ABCD parameters for Circuit – 24, and analyze voltage gain and impedance transformation. Where $R_1 = 100\ \Omega$, $R_2 = 220\ \Omega$, $R_3 = 1\text{ k}\Omega$ and $R_4 = 150\ \Omega$.

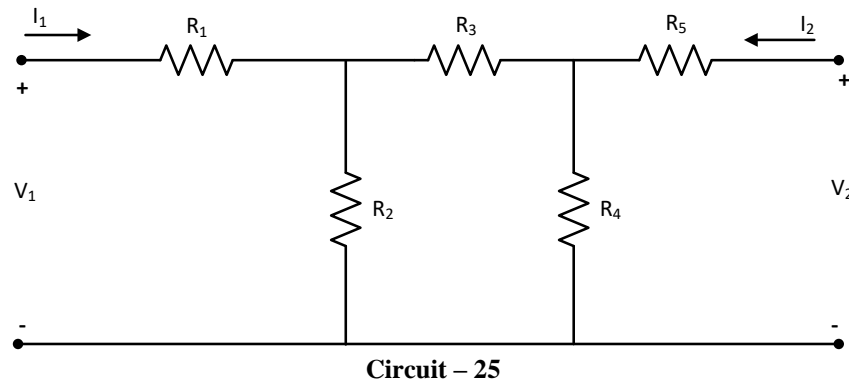


Circuit – 24

Try

1. Use Simulink to simulate Circuit – 24 for determining the H and ABCD parameters and compare the values to those you obtained from your experiment.
2. Calculate H parameters and convert in terms of Y parameters for Circuit – 25. Where $R_1 = 100\ \Omega$,

$R_2 = 220\ \Omega$, $R_3 = 1\text{ k}\Omega$, $R_4 = 150\ \Omega$ and $R_5 = 82\ \Omega$.



V. TEXT BOOKS:

1. A Chakrabarti, “*Circuit Theory*”, Dhanpat Rai Publications, 8th Edition, 2021.
2. William Hayt, Jack E Kemmerly S.M. Durbin, “*Engineering Circuit Analysis*”, Tata McGraw Hill, 9th Edition, 2020.

VI. REFERENCE BOOKS:

1. CL Wadhwa, *Electrical Circuit Analysis including Passive Network Synthesis*, International, 2nd Edition, 2009.
2. David A Bell, *Electric circuits*, Oxford University Press, 7th Edition, 2009.

VII. ELECTRONIC RESOURCES:

1. <https://www.allaboutcircuits.com/textbook/>
2. https://onlinecourses.nptel.ac.in/noc22_ee93/preview
3. <https://www.iare.ac.in>

VIII. MATERIALS ONLINE

12. Course template
13. Lab manual

COURSE CONTENT

ESSENTIALS OF INNOVATION								
I Semester: AE / ME / CE / ECE / EEE / CSE (AI&ML) / IT								
II Semester: CSE / CSE (DS) / CSE (CS)								
Course Code	Category	Hours / Week			Credits	Maximum Marks		
ACSD03	Foundation	L	T	P	C	CIA	SEE	Total
		-	2	-	1	40	60	100
Contact Classes: Nil	Tutorial Classes: Nil	Practical Classes: 45			Total Classes: 45			
Prerequisite: There are no prerequisites to take this course								

I. COURSE OVERVIEW:

Essentials of Innovation and Design thinking is a strategic approach towards creative problem-solving by placing users'/customers' needs above everything else. It is a process of questioning: questioning the problem, questioning assumptions, and questioning the implications. As a process it is a great catalyst of change and evolution. A Design thinking approach helps develop and build a culture of innovation across the students.

II. COURSE OBJECTIVES:

- I. The implications of disruption and the role of innovation.
- II. The various frameworks, tools, and techniques of design thinking.
- III. How to design, develop and implement an innovation product or service or process.

III. COURSE CONTENT:

Module-I; Philosophy of Innovation and Design Thinking

- Introduction to Innovation and Design Thinking
- History and Philosophy of Design Thinking
- Design Thinking as Problem-Solving Tool
- Design Thinking and it's Benefits
- Design Thinking Mind-set

Module-2: Mechanics of Innovation and Design Thinking

- Integrative View of Design Thinking
- Design Thinking Process
- 5 Stages (Empathise, Define, Ideate, Prototype and Test)
- Conceptual Frameworks Used in Design Thinking Process
- Case Studies

Module-3: Design Thinking for Understanding Customers

- Understanding the User and Context
- Market Research
- Visualization and Customer Journey Mapping
- Empathy Mapping
- Redefining Problems, Brainstorming
- Reframing the Perspectives
- Ideation and Creativity
- Creative Ideation Methodologies
- Sketching & Visualization
- Storytelling

Module-4: Implementing Design Thinking

- Innovating Products, Services and Business Models
- Concept Evaluation and Concept Development
- Applications of Design Thinking
- Designing for Tangibles and Intangibles
- Ideas and Opportunities for Products

Module-5: Innovation Management

- Introduction to Innovation Management
- Business, Product & Process Innovation
- Organization Innovation
- Innovating Products, Services and Business Models
- Crafting a Better World Using Design Thinking & Innovation
- Design Thinking, Innovation and Organization Strategy
- Idea Pitching and Validation

Text Books:

- I. Nigel Cross, “*Design Thinking: Understanding How Designers Think and Work*”, Kindle Edition, 2011.
- II. Tim Brown, Harper Bollins, “*Change by Design*”, 2009.
- III. Idris Mootee, “*Design Thinking for Strategic Innovation*”, John Wiley & Sons, 2013.

COURSE CONTENT

ENVIRONMENTAL SCIENCE								
I Semester: AE / ME / CE / ECE / EEE/ CSE (AI & ML) / CSE / CSE (CS) / CSE (DS) / IT								
Course Code	Category	Hours / Week			Credits	Maximum Marks		
AHSD06	Foundation	L	T	P	C	CIA	SEE	Total
		-	-	-	-	-	-	-
Contact Classes: Nil	Tutorial Classes: Nil	Practical Classes: Nil			Total Classes: Nil			
Prerequisite: Basic Principles of earth science.								

I. COURSE OVERVIEW:

This course is an interdisciplinary study which examines the interaction between humans and the environment, with specific reference to the effects of modern technological advances. The students will be able to understand the sustainable development, ecological sustainability, environmental pollution, environmental issues in order to protect the environment and followed by the application of this knowledge to current environmental problems in the later years.

II. COURSES OBJECTIVES:

The students will try to learn

- I. The interrelationship between living organism and environment.
- II. The importance of environment by assessing its impact on the human world
- III. The knowledge on themes of biodiversity, natural resources, pollution control and waste management.
- IV. The sustainability and unsustainability of various interactions between human society and the earth's natural systems

III. COURSE OUTCOMES:

At the end of the course students should be able to:

- CO1 Infer the basic ecological principles, biogeochemical cycles and its function for the flow of energy in ecosystem.
- CO2 Awareness on different natural resources and its conservation for sustainable development.
- CO3 Use alternate energy resources for future growing energy needs.
- CO4 Predict the of importance of biodiversity for its productive use.
- CO5 Identify the global environmental problems by different types of environmental Pollution and international summits for minimizing the problems.
- CO6 Outline the features of laws and rules related to environment protection, environmental impact assessment towards sustainable development.

IV. SYLLABUS:

MODULE-I: ECOSYSTEMS

Environment: definition, scope and importance of ecosystem, classification, structure and function of an ecosystem, food chains, food webs and ecological pyramids, flow of energy; biogeochemical cycles, hydrological cycle, phosphorous cycle, nitrogen cycle, biomagnifications.

MODULE-II: NATURAL RESOURCES

Natural resources: classification of resources, living and nonliving resources; water resources: use and over utilization of surface and ground water, floods and droughts, dams, benefits and problems; mineral resources: use and exploitation, environmental effects of extracting and using mineral resources; land resources; energy resources: renewable and non-renewable energy sources, use of alternate energy source.

MODULE-III: BIODIVERSITY AND BIOTIC RESOURCES

Biodiversity and biotic resources: introduction, definition, genetic, species and ecosystem diversity; value of biodiversity: consumptive use, productive use, social, ethical, aesthetic and optional values; Hot spots of biodiversity.

Threats to biodiversity: habitat loss, poaching of wildlife, human-wildlife conflicts; Conservation of biodiversity: In situ and ex situ conservation.

MODULE-IV: ENVIRONMENTAL POLLUTION AND CONTROL TECHNOLOGIES

Environmental pollution: definition, causes, effects and control measures of air pollution, water pollution, soil pollution, impacts of modern agriculture and noise pollution; solid waste: municipal solid waste management, composition and characteristics of e-waste and its management; Pollution control technologies: waste water treatment methods, primary, secondary and tertiary; global environmental issues and global efforts: climate change and impacts on human environment, ozone depletion, ozone depleting substances; International conventions / protocols: Kyoto protocol and Montreal protocol.

MODULE-V: ENVIRONMENTAL POLICY AND LEGISLATION

Environmental legislations: environmental protection act, air act 1981, water act, forest act. municipal solid waste management and handling rules, biomedical waste management and handling rules, hazardous waste management and handling rules, population and its explosion.

V. TEXT BOOKS:

1. Erach Bharucha, *Text Book of Environmental Studies for Under Graduate Course*, Orient Black Swan, 3rd Edition, 2021.
2. Anubha Kaushik and C P Kaushik, *Perspectives in Environmental Studies*, New Age International private limited, New Delhi, 7th Edition, 2021.
3. Benny Joseph, *Environmental Studies*, Tata Mc Graw Hill Publishing Co. Ltd, New Delhi, 3rd Edition, 2017.

VI. REFERENCE BOOKS:

1. Dr.M Anji Reddy, *Text Book of Environmental Science and Technology*, BS Publications, 3rd Edition, 2014.
2. Y Anjaneyulu, *Introduction to Environmental Science*, BSP Books Private Limited, 3rd Edition, 2020.

VII. ELECTRONICS RESOURCES:

1. <https://www.meripustak.com/Environmental-Science-Isv-8th-Edition-121505>
2. https://www.meripustak.com&gclid=CjwKCAjwtp2bBhAGEiWAOZZTuFwLEkGc6SGNUZjXpz0ffeNwgBOHWQIKge-E-9UvXxTPxQJdjaTgJB0CrQIQAvD_BwE

VIII. MATERIALS ONLINE

1. Course Template
2. Tutorial Question Bank
3. Model Question Paper – I
4. Model Question Paper - II
5. Lecture Notes
6. Early Lecture Readiness Videos

COURSE CONTENT

ENGINEERING CHEMISTRY								
I Semester: CSE / CSE (CS) / CSE (DS)								
II Semester: AE / ME / CE / ECE / EEE / CSE (AI & ML) / IT								
Course Code	Category	Hours / Week			Credits	Maximum Marks		
AHSD03	Foundation	L	T	P	C	CIA	SEE	Total
		3	-	-	3	40	60	100
Contact Classes: 64	Tutorial Classes: Nil	Practical Classes: Nil			Total Classes: 64			
Prerequisite: Basic principles of chemistry								

I. COURSE OVERVIEW:

This course focuses on the fundamental concepts of chemistry and then builds an interface with their industrial

applications. The basic knowledge on chemical bonding and intermolecular forces which together are responsible for determining the properties of materials. The students will be able to analyze water purification processes to avoid industrial interruptions. The course concludes with an overview of involving electron transfer, including their applications in corrosion and energy storage for portable electronic devices. It should cultivate in students to identify chemistry in each piece of finely engineered products used in households and industry.

II. COURSES OBJECTIVES:

The students will try to learn

- The concepts of electrochemical principles and causes of corrosion in the new developments and breakthroughs efficiently in engineering and technology.
- The different parameters to remove causes of hardness of water and their reactions towards complexometric method.
- The properties, separation techniques of natural gas and crude oil along with potential applications in major chemical reactions.
- The different types of materials with respect to mechanisms and its significance in industrial applications.

III. COURSE OUTCOMES:

At the end of the course students should be able to:

- CO1 Acquire the basic knowledge of electrochemical principles related to corrosion and its control
- CO2 Interpret the basic properties of water for its usage in industrial and domestic applications.
- CO3 Use complexometry for calculation of hardness of water to avoid industrial problems.
- CO4 Extend the applications of polymers based on their degradability and properties.
- CO5 Choose the appropriate fuel based on their calorific value for energy efficient processes.
- CO6 Predict the knowledge on viability of advanced materials for technological improvements in various sectors.

IV. COURSE CONTENT:

MODULE-I: BATTERIES CHEMISTRY AND CORROSION (10)

Introduction to electrochemical cells: Galvanic cell, electrolytic cell; electrochemical series and its applications; Batteries: classification of batteries, construction, working and applications of Zinc-air battery, Lead-acid battery, Li-ion battery, applications of Li-ion battery to electric vehicles; Corrosion: causes and effects of corrosion, theories of chemical and electrochemical corrosion, mechanism of electrochemical corrosion; Corrosion control methods: cathodic protection, sacrificial anode and impressed current methods; Metallic coatings: Galvanization and tinning, electroplating of Copper.

MODULE-II: WATER AND ITS TREATMENT (09)

Introduction: Hardness of water, causes of hardness; types of hardness, temporary and permanent hardness, expression and units of hardness; estimation of hardness of water by complexometric method; potable water and its specifications, steps involved in the treatment of water, disinfection of water by chlorination and ozonation; external treatment of water; ion-exchange process; desalination of water: reverse osmosis, numerical problems.

MODULE-III: POLYMER TECHNOLOGY (10)

Polymers: classification of polymers; types of polymerization-addition, condensation polymerization with examples. Plastics: thermoplastic and thermosetting plastics; preparation, properties and engineering applications of PVC, Nylon 6,6 and Bakelite.

Biodegradable polymers: polylactic acid and polyvinyl alcohol and their applications. Elastomers: Introduction to natural rubber, vulcanization of natural rubber, preparation, properties and engineering applications of Buna-S and Thiokol rubber.

MODULE-IV: ENERGY SOURCES (09)

Introduction to fuels; classification of fuels; Solid fuels: coal; analysis of coal, proximate and ultimate analysis and their significance; Liquid fuels: petroleum and its refining; Gaseous fuels: composition, characteristics and applications of natural gas, LPG and CNG; Alternative and non-conventional sources of energy: solar, wind and hydropower advantages and disadvantages. Calorific value of fuel: HCV and LCV, Dulong's formula, calculation of air quantity required for complete combustion of fuel, numerical problems.

MODULE-V: ENGINEERING MATERIALS (10)

Nanomaterials: Introduction, preparation of nanoparticles by sol-gel method, chemical reduction method

and applications of nanomaterials. Smart materials and their engineering applications: shape memory

materials, Poly L-Lactic acid. Thermoresponsive materials: Polyacryl amides, Poly vinyl amides.

Cement: composition of Portland cement, setting and hardening of cement.

Lubricants: characteristics of a good lubricant, mechanism of lubrication, thick film, thin film and extreme

pressure lubrication; properties of lubricants: viscosity, flash and fire point, cloud and pour point.

V. TEXT BOOKS:

1. JAIN & JAIN, P.C. Jain, Monika Jain, *Engineering Chemistry*, Dhanpat Rai publishing Company (P) limited, 17th edition, 2022.

VI. REFERENCE BOOKS:

1. Shashi Chawla, *Text Book of Engineering Chemistry*, Dhanat Rai and Company (P) Limited, 1st Edition, 2017.
2. Jayashree Anireddy, *Textbook of Engineering chemistry*, Wiley Publications, 2023
3. S.S.Dara, *Text of Engineering Chemistry*, S.Chand & Co, New Delhi, 12th edition, 2018.
4. Nitin K Puri, *Nanomaterials Synthesis Properties and Applications*, I K international publishing house pvt Ltd, 1st edition 2021.

VII. ELECTRONIC RESOURCES:

1. Engineering chemistry (NPTEL Web-book), by B. L. Tembe, Kamaluddin and M. S.Krishnan.http://www.cdeep.iitb.ac.in/webpage_data/nptel/Core%20Science/Engineering%20Chemistry%201/About-Faculty.html
2. https://books.google.co.in/books?id=R1JtyILNIsAC&pg=PR3&source=gbs_selected_page&cad=3#v=onepage&q&f=false
3. https://books.google.co.in/books?id=eQTLcGAAQBAJ&pg=SA1PA53&source=gbs_selected_pages&cad=3#v=onepage&q&f=false

VIII. MATERIALS ONLINE

1. Course Template
2. Tutorial Question Bank
3. Tech talk Topics
4. Assignments
5. Definition and Terminology
6. Model Question Paper – I
7. Model Question Paper - II
8. Lecture Notes
9. Early Lecture Readiness Videos
10. Power point presentation

COURSE CONTENT

APPLIED PHYSICS								
I Semester: CSE / CSE (CS) / CSE (DS)								
II Semester: AE / ME / CE / ECE / EEE / CSE (AI & ML) / IT								
Course Code	Category	Hours / Week			Credits	Maximum Marks		
AHSD07	Foundation	L	T	P	C	CIA	SEE	Total
		3	-	-	3	40	60	100
Contact Classes: 48	Tutorial Classes: Nil	Practical Classes: Nil			Total Classes: 48			
Prerequisite: Basic principles of physics								

I. COURSE OVERVIEW:

The aim of this course is to enhance understanding of fundamental knowledge in physics needed for the future technological advances. The framework prepares students to engage in scientific questioning and extend thinking to investigations. The concepts cover current topics in the fields of solid state physics, modern physics, superconductors and nanotechnology. This knowledge helps to develop the ability to apply the principles in many technological sectors such as nanotechnology, optical fiber communication, quantum technology etc.

II. COURSES OBJECTIVES:

The students will try to learn

- V. Fundamental concepts needed to explain a crystal structure in terms of atom positions, unit cells, and crystal symmetry.
- VI. Basic formulations in wave mechanics for the evolution of energy levels and quantization of energies for a particle in a potential box with the help of mathematical description.
- VII. The metrics of optoelectronic components, lasers, optical fiber communication and be able to incorporate them into systems for optimal performance.
- VIII. The appropriate magnetic, superconducting and nanomaterials required for various engineering applications.

III. COURSE OUTCOMES:

At the end of the course students should be able to:

- CO 1 Use the general rules of indexing of directions and planes in lattices to identify the crystal systems and the Bravais lattices.
- CO 2 Use the concepts of dual nature of matter and Schrodinger wave equation to a particle enclosed in simple systems.
- CO 3 Analyze the concepts of laser with normal light in terms of mechanism for applications in different fields and scientific practices.
- CO 4 Strengthen the knowledge on functionality of components in optical fiber communication system by using the basics of signal propagation, attenuation and dispersion.
- CO 5 Gain deeper understanding on properties of magnetic and superconducting materials suitable for engineering applications.
- CO 6 Review the principle factors, fabrication, characterization techniques and the applications of nanomaterials.

IV. COURSE CONTENT:

MODULE - I: CRYSTAL STRUCTURES (10)

Introduction, space lattice, basis, unit cell, lattice parameter, Bravais lattices, crystal systems, structure and packing fractions of simple cubic, body centered cubic, face centered cubic crystals, directions and planes in crystals, Miller indices, separation between successive [h k l] planes.

MODULE –II: QUANTUM PHYSICS (09)

Waves and particles, de Broglie hypothesis, matter waves, Davisson and Germer's experiment, Schrödinger's time independent wave equation, physical significance of the wave function, infinite square well potential.

MODULE –III: LASERS AND FIBER OPTICS (10)

Characteristics of lasers, spontaneous and stimulated emission of radiation, population inversion, lasing action, Ruby laser, He-Ne laser, applications of lasers.

Principle and construction of an optical fiber, acceptance angle, numerical aperture, types of optical fibers

(Single mode, multimode, step index, graded index), optical fiber communication system with block diagram, applications of optical fibers.

MODULE –IV: MAGNETIC AND SUPERCONDUCTING PROPERTIES (10)

Permeability, field intensity, magnetic field induction, magnetization, magnetic susceptibility, origin of magnetic moment, Bohr magneton, classification of dia, para and ferro magnetic materials on the basis of magnetic moment, Hysteresis curve.

Superconductivity, general properties, Meissner effect, effect of magnetic field, type-I & type-II superconductors, BCS theory, applications of superconductors.

MODULE –V: NANOTECHNOLOGY (09)

Nanoscale, quantum confinement, surface to volume ratio, bottom-up fabrication: Sol-gel, precipitation, combustion methods, top-down fabrication: Ball milling, physical vapor deposition, chemical vapor deposition, characterization techniques: X-ray diffraction, transmission electron microscopy, applications of nanomaterials.

V. TEXT BOOKS:

2. Arthur Beiser, Shobhit Mahajan and Rai Choudhary, *Concepts of Modern Physics*, Tata McGraw Hill, 7th Edition, 2017.

VI. REFERENCE BOOKS:

2. H J Callister, *A Textbook of Materials Science and Engineering*, Wiley Eastern Edition, 8th Edition, 2013.
3. Halliday, Resnick and Walker, *Fundamentals of Physics*, John Wiley & Sons, 11th Edition, 2018.
4. Charles Kittel, *Introduction to Solid State Physics*, Wiley Eastern, 2019.
5. S.L. Gupta and V. Kumar, *Elementary Solid State Physics*, Pragathi Prakashan, 2019.
6. K K Chattopadhyay and A N Banerjee, *Introduction to Nanoscience and Nanotechnology*, Prentice Hall India, 2nd Edition, 2011.

VII. ELECTRONIC RESOURCES:

4. NPTEL :: Physics - NOC: Quantum Mechanics I
5. NPTEL :: Physics - NOC: Introduction to Solid State Physics
6. NPTEL :: Physics - NOC: Solid State Physics
7. <https://nptel.ac.in/courses/104104085>
8. NPTEL :: Metallurgy and Material Science - NOC: Nanotechnology, Science and Applications

VIII. MATERIALS ONLINE

11. Course template
12. Tutorial question bank
13. Definition and terminology
14. Tech-talk topics
15. Assignments
16. Model question paper - I
17. Model question paper - II
18. Lecture notes
19. Early learning readiness videos (ELRV)
20. Power point presentations

COURSE CONTENT

DIFFERENTIAL EQUATIONS AND VECTOR CALCULUS								
II Semester: AE / ME / CE / ECE / EEE / CSE / CSE (AI & ML) / CSE (DS) / CSE (CS) / IT								
Course Code	Category	Hours/Week			Credits	Maximum Marks		
AHSD08	Foundation	L	T	P	C	CIA	SEE	Total
		3	1	-	4	40	60	100
Contact Classes: 48	Tutorial Classes: 16	Practical Classes: Nil			Total Classes: 64			
Prerequisite: Basic Principles of Matrices and Calculus								

I. COURSE OVERVIEW:

This course serves as a foundation course on differential equations and vector calculus. It includes techniques for solving ordinary differential equations, partial differential equations, vector differentiation and vector integration. It is designed to extract the mathematical developments, skills, from basic concepts to advance level of engineering problems to meet the technological challenges.

II. COURSE OBJECTIVES:

The students will try to learn:

- I The analytical methods for solving first and higher order differential equations with constant coefficients.
- II The analytical methods for formation and solving partial differential equations.
- III The physical quantities of vector valued functions involved in engineering field.
- IV The logic of vector theorems for finding line, surface and volume integrals.

III. COURSE OUTCOMES:

At the end of the course students should be able to:

- CO1 Utilize the methods of differential equations for solving the orthogonal trajectories and Newton's law of cooling.
- CO2 Solve the higher order linear differential equations with constant coefficients by using method of variation of parameters.
- CO3 Make use of analytical methods for PDE formation to solve boundary value problems.
- CO4 Identify various techniques of Lagrange's method for solving linear partial differential equations which occur in science and engineering.
- CO5 Interpret the vector differential operators and their relationships for solving engineering problems.
- CO6 Apply the integral transformations to surface, volume and line of different geometrical models in the domain of engineering.

IV. COURSE CONTENT:

MODULE-I: FIRST ORDER AND FIRST DEGREE ODE (10)

Exact differential equations, Equations reducible to exact differential equations, linear and Bernoulli's equations, Applications: Orthogonal Trajectories (Cartesian Coordinates) Newton's law of cooling.

MODULE-II: ORDINARY DIFFERENTIAL EQUATIONS OF HIGHER ORDER (10)

Second order linear differential equations with constant coefficients: non-homogeneous terms of the type e^{ax} , $\sin ax$, $\cos ax$, polynomials in x , $e^{ax}(x)$ and method of variation of parameters.

MODULE-III: PARTIAL DIFFERENTIAL EQUATIONS (09)

Formation of partial differential equations by elimination of arbitrary constants and arbitrary functions.

Solutions of first order linear equations: method of grouping and method of multipliers.

MODULE–IV: VECTOR DIFFERENTIATION (09)

Scalar and vector point functions; definitions of gradient, divergent and curl with examples; solenoidal and irrotational vector point functions; scalar potential function.

MODULE–V: VECTOR INTEGRATION (10)

Line integral, surface integral and volume integral, Green's theorem in a plane, Stoke's theorem and Gauss divergence theorem without proofs.

V. TEXT BOOKS:

1. B. S. Grewal, *Higher Engineering Mathematics*, 44/e, Khanna Publishers, 2017.
2. Erwin Kreyszig, *Advanced Engineering Mathematics*, 10/e, John Wiley & Sons, 2011.

VI. REFERENCE BOOKS:

1. R. K. Jain and S. R. K. Iyengar, *Advanced Engineering Mathematics*, 3/ed, Narosa Publications, 5th Edition, 2016.
2. George B. Thomas, Maurice D. Weir and Joel Hass, Thomas, *Calculus*, 13/e, Pearson Publishers, 2013.
3. N.P.Bali and Manish Goyal, *A text book of Engineering Mathematics*, Laxmi Publications, Reprint, 2008
4. Dean G. Duffy, *Advanced Engineering Mathematics with MATLAB*, CRC Press.
5. Peter O'Neil *Advanced Engineering Mathematics*, Cengage Learning.
6. B.V. Ramana, *Higher Engineering Mathematics*, McGraw Hill Education.

VII. ELECTRONIC RESOURCES:

1. Engineering Mathematics - I, By Prof. Jitendra Kumar
https://onlinecourses.nptel.ac.in/noc23_ma88/preview
2. Advanced Calculus for Engineers, By Prof. Jitendra Kumar, Prof. Somesh Kumar
https://onlinecourses.nptel.ac.in/noc23_ma86/preview
3. http://www.efunda.com/math/math_home/math.cfm
4. <http://www.ocw.mit.edu/resources/#Mathematics>
5. <http://www.sosmath.com>
6. <http://www.mathworld.wolfram.com>

VIII. MATERIAL ONLINE:

1. Course template
2. Tech-talk topics
3. Assignments
4. Definition and terminology
5. Tutorial question bank
6. Model question paper – I
7. Model question paper – II
8. Lecture notes
9. Early lecture readiness videos (ELRV)
10. Power point presentations

COURSE CONTENT

ESSENTIALS OF PROBLEM SOLVING								
II Semester: CSE / CSE (AI & ML) / CSE (DS) / CSE (CS) / IT / ECE / EEE								
Course Code	Category	Hours / Week			Credits	Maximum Marks		
ACSD05	Foundation	L	T	P	C	CIA	SEE	Total
		3	0	0	3	40	60	100
Contact Classes: 48	Tutorial Classes: Nil	Practical Classes: Nil			Total Classes: 48			
Prerequisite: There is no prerequisite to take this course								

I. COURSE OVERVIEW:

This course aims to provide exposure to problem solving through programming. Useful graph theory concepts, numerical techniques, and their applications to real world problems are discussed. Graph theoretical notions and the use of algorithms, both in the mathematical theory of graphs and its applications are discussed. Student will also learn how to implement and interpret numerical solutions by writing a well-designed computer programs in regard to their efficiency and suitability for real-life applications.

II. COURSES OBJECTIVES:

The students will try to learn

- I. The fundamental concepts of graph theory and its properties.
- II. The basics related to paths and cycles using Eulerian and Hamiltonian cycles.
- III. The applications of graph colouring and traversal algorithms for solving real-time problems.
- IV. The numerical methods to solve algebraic equations.

III. COURSE OUTCOMES:

At the end of the course students should be able to:

- CO1 Outline the graph terminologies, graph representation, and relate them to practical examples.
- CO2 Build efficient graph routing algorithms for various optimization problems on graphs.
- CO3 Use effective techniques from graph theory to solve problems in networking and telecommunication.
- CO4 Interpret the fundamental concepts of polynomials, roots of equations and solve corresponding problems using computer programs.
- CO5 Apply the knowledge of numerical methods to solve algebraic and transcendental equations arising in real-life situations.
- CO6 Solve numerical integrals and ordinary differential equations to simulate discrete time algorithms.

IV. COURSE CONTENT:

MODULE - I GRAPH THEORY (08)

Graph terminology, digraphs, weighted graphs, complete graphs, graph complements, bipartite graphs, graph combinations, isomorphisms, matrix representations of graphs, incidence and adjacency matrices, degree sequence.

MODULE - II GRAPH ROUTES (10)

Eulerian circuit: Königsberg bridge problem, touring a graph; Eulerian graphs, Hamiltonian cycles, the traveling salesman problem; Shortest paths: Dijkstra's algorithm, walks using matrices.

MODULE - III GRAPH COLORING AND GRAPH ALGORITHMS (10)

Four color theorem, vertex coloring, edge coloring, coloring variations, first-fit coloring algorithm.

Graph traversal: depth-first search, breadth-first search and its applications; Minimum spanning trees: Kruskal's and Prim's algorithm, union-find structure.

MODULE - IV: ALGEBRAIC AND TRANSCENDENTAL EQUATIONS (10)

Algebraic equations, method of false position, bisection method, iteration method, Newton-Raphson method, Secant method, Ramanujan's Method, Muller's method (Approximation up to 2 decimals only).

MODULE - V: NUMERICAL INTEGRATION AND ORDINARY DIFFERENTIAL EQUATIONS (10)

Trapezoidal rule, Simpson's 1/3 rule, Simpson's 3/8 rule, Solution by Taylor's series, Euler's method of solving an ordinary differential equation numerically, Runge-Kutta's second order method of solving ordinary differential equations (Approximation up to 2 decimals only).

V. TEXT BOOKS:

2. Karin R Saoub, *Graph Theory: An Introduction to Proofs, Algorithms, and Applications*, 1st edition, Chapman and Hall, 2021.
3. S S Sastry, *Introductory Methods of Numerical Analysis*, 5th edition, 2012.

VI. REFERENCE BOOKS:

1. Mahinder Kumar Jain, *Numerical Methods: For Scientific and Scientific Computation*, New Age International Pvt. Ltd., 7th edition, 2019.
2. P Kandasamy, K Thilagavathy, K Gunavathi, *Numerical Methods*, S Chand and Company, 2006.
3. R Balakrishnan, K Ranganathan, *A Textbook of Graph Theory*, Springer Exclusive, 2nd edition, 2019.
4. Jann Kiusalaas, *Numerical Methods in Engineering with Python*, Cambridge University Press, 2nd edition 2010.
5. Gary Chartrand, Ping Zhang, *A First Course in Graph Theory*, Dover Publications Inc., 2012.
6. James F. Epperson, *An Introduction to Numerical Methods and Analysis*, Wiley, 2nd edition, 2013.

VII. ELECTRONIC RESOURCES:

1. <https://www.geeksforgeeks.org/numerical-methods-and-calculus-gq/>
2. <https://www.geeksforgeeks.org/program-for-bisection-method/>
3. <https://ocw.mit.edu/courses/2-993j-introduction-to-numerical-analysis-for-engineering-13-002j-spring-2005/pages/lecture-notes/>
4. <https://www.tutorialspoint.com/graphs-and-its-traversal-algorithms>
5. https://web.mit.edu/urban_or_book/www/book/chapter6/6.4.4.html
6. <https://www.hackerearth.com/practice/algorithms/graphs/minimum-spanning-tree/tutorial/>
7. <https://www.codingninjas.com/studio/library/euler-and-hamilton-paths>

VIII. MATERIALS ONLINE

1. Course template
2. Tutorial question bank
3. Tech-talk topics
4. Open-ended experiments
5. Definitions and terminology
6. Assignments
7. Model question paper – I
8. Model question paper – II
9. Lecture notes
10. PowerPoint presentation
11. E-Learning Readiness Videos (ELRV)

COURSE CONTENT

ENGINEERING CHEMISTRY LABORATORY								
I Semester: CSE / CSE (CS) / CSE (DS)								
II Semester: AE / ME / CE / ECE / EEE / CSE (AI&ML) / IT								
Course Code	Category	Hours / Week			Credits	Maximum Marks		
AHSD05	Foundation	L	T	P	C	CIA	SEE	Total
		-	-	3	1	40	60	100
Contact Classes: Nil	Tutorial Classes: Nil	Practical Classes: 36			Total Classes: 36			
Prerequisite: Basic Principles of Chemistry								

I. COURSE OVERVIEW:

The course encourages introducing analytical tools in an Engineering perspective. The course efforts to provide the basic knowledge of analytical methodology, outlines the importance of volumetric analysis, comprehensive instrumental analysis for properties of fuels, colorimetric analysis and spectroscopic analysis. This practical approach gives the essence of analytical chemistry for skill development in determinations of materials properties and its viability in the industry.

II. COURSES OBJECTIVES:

The students will try to learn:

- I. The quantitative analysis to know the strength of unknown solutions by instrumental methods.
- I. The troubles of hard water and its estimation by analytical techniques.
- II. The applications of appropriate lubricant for finely tuned machinery.
- III. The basic knowledge on quantity of light absorbed by the materials.

III. COURSE OUTCOMES:

After successful completion of the course students should be able to:

- CO1 Use conductivity meter and potentiometer for measurement of conductance and electromotive force of solutions
- CO2 Examine the corrosion tendency in metals and its control by using inhibitor.
- CO3 Make use of the principles of water analysis for domestic and industrial applications.
- CO4 Demonstrate the characteristics of different fuels for finding the calorific value.
- CO5 Use different types of lubricants to know its properties for the proper lubrication of machinery in industries.
- CO6 Interpret the absorption tendency of solids or liquids by using colorimetry and spectroscopy techniques.

IV. COURSE CONTENT:

1. GETTING STARTED EXERCISES

1.1 Introduction to Chemistry Laboratory

The fundamental concepts and theories required for carrying out qualitative and quantitative analysis.

Detailed explanation on the analytical techniques used for qualitative analysis. Emphasis on instrumental method of analysis and its advantages over conventional methods.

- i. Types of analysis
 - ii. Difference between qualitative and quantitative analysis
 - iii. Common techniques of qualitative and quantitative analysis
 - iv. Introduction to instrumental method of analysis
 - v. Introduction to basic techniques and handling of common apparatus
 - vi. Discussion of Material Safety Data Sheet (MSDS) of chemicals
 - vii. Identification of toxic signs and safety procedures of chemical laboratory
-

1.2 Safety Guidelines to Chemistry Laboratory

The chemistry laboratory must be a safe place in which to work and learn about chemistry.

- i. Wear a chemical-resistant apron.
 - ii. Be familiar with your lab work sheet before you come to lab. Follow all written and verbal instructions carefully. Observe the safety alerts in the laboratory directions. If you do not understand a direction or part of a procedure, ask the teacher before proceeding.
 - iii. When entering the laboratory room, do not touch any equipment, chemicals, or other materials without being instructed to do so. Perform only those experiments authorized by the instructor.
 - iv. If you take more of a chemical substance from a container than you need, you should not return the excess to the container. This might cause contamination of the substance remaining. Dispose of the excess as your instructor directs.
 - v. Never smell anything in the laboratory unless your teacher tells you it is safe. Do not smell a substance by putting your nose directly over the container and inhaling. Instead, waft the vapors toward your nose by gently fanning the vapors toward yourself.
 - vi. Do not directly touch any chemical with your hands. Never taste materials in the laboratory.
 - vii. Work areas should be kept clean and tidy at all times. Always replace lids or caps on bottles and jars.
-

1.3 Data recording and reports

Students must record their experimental values in the provided tables in this laboratory manual and reproduce them in the laboratory worksheets. Worksheets are integral to recording the methodology and results of an experiment. In engineering practice, the laboratory worksheets serve as a valuable reference to the technique used in the laboratory. Note that the data collected will be an accurate and permanent record of the data obtained during the experiment and the analysis of the results.

2. CONDUCTOMETRY

2.1 Determine the neutralization point between strong acid against strong base

- i. **The basic principle of conductometric titrations**
 - ii. **Titration of unknown solution of acid with base**
 - iii. **Graphical plots on volume of titrant vs. conductance**
-

3. POTENTIOMETRY

3.1 Estimate the amount of Iron by using potentiometry

- i. **The basic principle of potentiometric titrations**
 - ii. **Titration of Mohr's salt with potassium dichromate**
 - iii. **Graphical plots on volume of titrant vs. potential**
-

4. pH METRY

4.1 Determine the pH of the unknown solution by pH metry

- i. **The basic principle of pH metry**
 - ii. **Titration of unknown solution with standard acid**
 - iii. **Graphical plots on volume of titrant vs. pH to obtain equivalence point**
-

5. ARGENTOMETRIC TITRATIONS

5.1 Determination of chloride content of water by argentometry

- i. **Principle of Argentometric titration.**
 - ii. **Titration of water samples by using EDTA to find the total hardness in water.**
-

6. MEASUREMENT OF TOTAL DISSOLVED SOLIDS IN WATER

6.1 Measurement of total dissolved solids (TDS) in different water samples

- i. **Specifications of potable water**
 - ii. **Measure the total dissolved solids in different water samples by TDS meter**
-

7. COMPLEXOMETRY METHOD

7.1 Estimate the total hardness of water by EDTA

- i. **Principle of complexometric titration**
 - ii. **Titration of water samples by using EDTA to find the total hardness in water.**
-

8. BOMB CALORIMETER

8.1 Determine the calorific value of solid or liquid fuels using Bomb calorimeter

- i. Significance of bomb calorimeter
- ii. Combustion of solid or liquid fuels in bomb calorimeter to find its calorific value

9. VISCOSITY OF LUBRICANT

9.1 Determine the viscosity of the lubricants using Red Wood viscometer / Ostwald's viscometer

- i. The principle of viscosity of lubricant
- ii. Significance of viscosity index of lubricant
- iii. Viscosity of given lubricant at various temperature by using Red wood viscometer

10. FLASH AND FIRE POINTS OF LUBRICANT

10.1 Determine the flash and fire points of lubricants

- i. Significance of flash and fire point of lubricant in industries
- ii. Flash and Fire points of a given lubricant by using Pensky Martens flash point apparatus

11. CLOUD AND POUR POINTS OF LUBRICANT

11.1 Determine cloud and pour points of lubricants

- i. Significance of cloud and pour point of lubricants in industries
- ii. Cloud and Pour points of a given lubricant by using cloud and pour point apparatus

12. COLORIMETRY

12.1 Estimate the metal ion concentration using colorimeter

- i. Complexation of metal ion with ligands
- ii. Detection of absorbance of the colored metal -ligand complex solution
- iii. Graphical determination of concentration of the metal ions in the solution

13. SPECTROSCOPY

13.1 Characterization of nanomaterials by UV-visible spectrophotometer

- i. Synthesis of silver oxide nanomaterials
- ii. Dispersion of nanoparticles in suitable solvent
- iii. Determination of absorption edge of the nanoparticles using spectroscopic technique

V. TEXTBOOKS:

1. K. Mukkanti et al. *Practical Engineering Chemistry*, B.S. Publications, Hyderabad.
2. Vogel's, *Quantitative chemical analysis*, prentice Hall, 6th Edition, 2009.

VI. REFERENCE BOOKS:

1. Solanki, M. K. *Engineering Chemistry Laboratory Manual*. (Edu creation Publishing, 2019).
2. Jeffery, G. H. in *TEXTBOOK OF QUANTITATIVE CHEMICAL ANALYSIS* (ed John Wiley and Sons) (1989).
3. Gary-D-Christian, P. K. S. D., Kevin A. Schug. *Analytical-Chemistry-by-Gary-D-Christian*. 7 edn, Vol. 7 826 (Wiley, 2014).
4. Budinski, Kenneth G., *Engineering materials: properties and selection*, 5th edition, Prentice-Hall, 1996, pg.423.
5. *Engineering chemistry by Jain & Jain*, 17th Edition, ISBN:978-93-5216-641-1
6. Nitin K Puri, "Nanomaterials synthesis properties and applications" I K international publishing house Pvt Ltd, 1st Edition 2021.

7. B. Ramadevi and P. Aparna, S Chand publications, *lab manual for engineering chemistry*, S Chand publications, NewDelhi, 1st Edition 2022.

VII. ELECTRONIC RESOURCES:

1. <https://nptel.ac.in/translation>
2. <https://nptel.ac.in/courses/115105120>
3. <https://archive.nptel.ac.in/courses/122/101/122101001/#>

VIII. MATERIALS ONLINE

1. Course Template
2. Laboratory Manual

COURSE CONTENT

APPLIED PHYSICS LABORATORY								
I Semester: CSE / CSE (CS) / CSE (DS)								
II Semester: AE / ME / CE / ECE / EEE / CSE (AI & ML) / IT								
Course Code	Category	Hours / Week			Credits	Maximum Marks		
AHSD09	Foundation	L	T	P	C	CIA	SEE	Total
		-	-	2	1	40	60	100
Contact Classes: Nil	Tutorial Classes: Nil	Practical Classes: 45			Total Classes: 45			
Prerequisite: Basic Principles of Physics								

I. COURSE OVERVIEW:

The aim of the course is to provide hands on experience for experiments in different areas of physics. Students will be able to perform the experiments with interest and an attitude of learning. This laboratory includes experiments involving electromagnetism and optoelectronics. These also develop student's expertise in applying physical concepts to practical problem and apply it for different applications.

II. COURSES OBJECTIVES:

The students will try to learn:

- I Familiarize with the lab facilities, equipment, standard operating procedures.
- II The different kinds of functional magnetic materials which paves a way for them to use in various technical and engineering applications.
- III The analytical techniques and graphical analysis to study the experimental data for optoelectronic devices.
- IV The application of characteristics of lasers and its propagation in optical fiber communication.

III. COURSE OUTCOMES:

After successful completion of the course, students should be able to:

- CO1 Identify the type of semiconductor using the principle of Hall effect and also determine the energy gap and resistivity of a semiconductor diode using four probe methods.
- CO2 Illustrate principle, working and application of wave propagation and compare the results of frequency with theoretical harmonics and overtones.
- CO3 Investigate the energy losses, curie temperature and properties associated with a given Ferro magnetic material.
- CO4 Examine launching of light through optical fiber from the concept of light gathering capacity of numerical aperture and determine the divergence of Laser beam
- CO5 Graph V-I /L-I characteristics of various optoelectronic devices like Light Emitting diode, Solar cell at different intensities to understand their basic principle of functioning as well as to infer the value of Planck's constant.
- CO6 Analyze the variation of magnetic field induction produced at various points along the axis of current carrying coil.

IV. COURSE CONTENT:

1. GETTING STARTED EXERCISES

1.1 On Errors and Uncertainty in a measurement:

When a number represents a physical measurement, it is never exact because of the limitations of the instrument used or the way it was employed etc. It is essential, therefore, that each experimental result be presented in a way that indicates its reliability. The accuracy of result is important, for example, the calibration of the measuring instruments or systematic errors on the part of whoever is taking the data.

The following table is useful in thinking about these concepts:

Problem	Remedy
Mistakes and blunders	Repeat measurements several times to check yourself
Systematic errors	Use calibrated instruments properly and carefully
Random errors	Treat data statistically and report on the average magnitude of errors

1.2 Making a good graph:

- Keep your axes straight: If you need to plot "A vs B", or "A as a function of B", then A is on the vertical axis and B is on the horizontal axis.
 - The crucial part is choosing the range and scale for each axis. The range must be just large enough to accommodate all data and small enough that the scale is readable.
 - The scale should be spread out enough so that data take up most of the graph area and labeled so that plotting (and reading) is easy. Where appropriate, error bars should be included to indicate the uncertainty in measurements.
 - When a line is drawn, it should be a smooth one that best fits data. In general, there should be as many points on one side of the line as on the other. If data is taken properly, the line should pass inside of the error bars for each point.
 - If graph shows that one quantity is proportional to another, it should be a straight line that starts at the origin and passes through the plotted data with as many points on one side as the other.
 - If the slope of the line is to be found, choose two points on the line that are as far apart as possible. This will minimize the error that is introduced in reading the value of those points.
 - The slope is the difference between the vertical values of those points divided by the difference in the horizontal values of those points.
-

1.3 Data recording and worksheets

- Write the work sheets for the allotted experiment and keep them ready before the beginning of each lab.
- Perform the experiment and record the observations in the worksheets.
- Analyze the results and get the work sheets evaluated by the faculty.
- Upload the evaluated reports online from CMS LOGIN within the stipulated time.

2. HALL EFFECT (LORENTZ FORCE)

- 2.1 Study the phenomenon of Hall effect and determine the charge carrier density and Hall coefficient of a given sample.
- 2.2 Hint whether the given semiconductor is p - type or n - type using the principle of Hall Effect.

ENERGY GAP OF A SEMICONDUCTOR DIODE

- 2.3 Determination of energy gap of a given semiconductor diode by measuring the variation of current as a function of temperature.
 - 2.4 Try to find the Fermi level of the given semiconductor
-

3. RESISTIVITY – FOUR PROBE METHOD

- 3.1 Determination of the resistivity by forcing current through two outer probes
 - 3.2 Formulate the reading of voltage across the two inner probes of semiconductor by four probe method.
-

4. MELDE'E EXPERIMENT

- 4.1 Determination of frequency of a given tuning fork in longitudinal wave propagation.
 - 4.2 Try to establish the transverse mode of wave propagation by understanding the theoretical harmonics and overtones.
-

5. B-H CURVE WITH CRO

- 5.1 Evaluate the energy loss per unit volume of a given magnetic material per cycle by tracing the hysteresis loop (B-H curve).
 - 5.2 Observe the hysteresis loss of ferro magnetic materials.
-

6. MAGNETIC MATERIAL

- 6.1 Determine the curie temperature (T_c) of a ferromagnetic material.
 - 6.2 Evaluate the relative permeability (μ_r) of a ferromagnetic material.
-

7. OPTICAL FIBER

- 7.1 Determine the numerical aperture of a given optical fiber.
 - 7.2 Calculate the acceptance angle of a given optical fiber.
-

8. LASER DIVERGENCE

- 8.1 Determination of the beam divergence of the given laser beam.
 - 8.2 Try to estimate the laser output
-

SOLAR CELL

- 8.3 Studying the characteristics of solar cell at different intensities
 - 8.4 Try to get the maximum workable power.
-

9. LIGHT EMITTING DIODE

- 9.1 Studying V-I characteristics of LED in forward bias for different LEDs.
 - 9.2 Measure the threshold voltage and forward resistance, and try for the dynamic Resistance
-

PLANCK'S CONSTANT

- 9.3 Determination of Planck's constant by measuring threshold voltage of given LED.
 - 9.4 Draw the L -I characteristics of the given LED.
-

10. STEWART GEE'S APPARATUS

- 10.1 Study the magnetic field along the axis of current carrying coil – Stewart and Gee's method.
- 10.2 Estimate the magnetic lines of force.

11. BIASING OF DIODE

- 11.1 Study the forward bias of LED
- 11.2 Study the reverse bias of Photo diode

V. TEXT BOOKS:

1. Laboratory Experiments in College Physics", C.H. Bernard and C.D. Epp, John Wiley and Sons, Inc., New York, 1995.

VI. REFERENCE BOOKS:

1. C. L. Arora, "Practical Physics", S. Chand & Co., New Delhi, 3rd Edition, 2012.
2. Vijay Kumar, Dr. T Radhakrishna, "Practical Physics for Engineering Students", SM Enterprises, 2nd Edition, 2014.
3. Dr. Rizwana, "Engineering Physics Manual", Spectrum Techno Press, 2018.

VII. ELECTRONIC RESOURCES:

4. <https://nptel.ac.in/translation>
5. <https://nptel.ac.in/courses/115105120>
6. NPTEL:: courses-Sem 1 and 2 - Engineering Physics and Applied Physics I
7. [Experimental Physics I - Course \(nptel.ac.in\)](https://nptel.ac.in)
8. [NPTEL:: Physics - Waves and Oscillations](https://nptel.ac.in)

VIII. MATERIALS ONLINE:

1. Course template
2. Lab manual



INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal - 500 043, Hyderabad, Telangana

COURSE CONTENT

PROGRAMMING FOR PROBLEM SOLVING LABORATORY								
II Semester: AE / ME / CE / ECE / EEE / CSE / CSE (AI & ML) / CSE (DS) / CSE (CS) / IT								
Course Code	Category	Hours / Week			Credits	Maximum Marks		
ACSD06	Foundation	L	T	P	C	CIA	SEE	Total
		0	1	2	2	40	60	100
Contact Classes: Nil	Tutorial Classes: 15	Practical Classes: 30			Total Classes: 45			
Prerequisites: There are no prerequisites to take this course.								

I. COURSE OVERVIEW:

The course is designed with the fundamental programming skills and problem-solving strategies necessary to tackle a wide range of computational challenges. Through hands-on programming exercises and projects, students will learn how to write code, analyze problems and develop solutions using various programming languages and tools. The course will cover fundamental programming concepts and gradually progress to more advanced topics.

II. COURSE OBJECTIVES

The students will try to learn:

- The fundamental programming constructs and use of collection data types in Python.
- The ability to develop programs using object-oriented features.
- Basic data structures and algorithms for efficient problem-solving.
- Principles of graph theory and be able to apply their knowledge to a wide range of practical problems across various disciplines.

III. COURSE OUTCOMES:

At the end of the course students should be able to:

- CO1 Adapt programming concepts, syntax, and data structures through hands on coding exercises.
- CO2 Develop the ability to solve a variety of programming problems and algorithms using python.
- CO3 Implement complex and custom data structures to solve real-world problems.
- CO4 Demonstrate proficiency in implementing graph algorithms to solve variety of problems and scenarios.
- CO5 Develop critical thinking skills to solve the various real-world applications using graph theory.
- CO6 Learn the importance of numerical methods and apply them to tackle a wide range of computational problems.

IV. COURSE CONTENT:

EXERCISES FOR PROGRAMMING FOR PROBLEM SOLVING LABORATORY

Note: Students are encouraged to bring their own laptops for laboratory practice sessions.

1. Getting Started Exercises

1.1 Two Sum

Given an array of integers `nums` and an integer `target`, return indices of the two numbers such that they add up to `target`. You may assume that each input would have exactly one solution, and you may not use the same element twice. You can return the answer in any order.

Input: `nums = [2, 7, 11, 15]`, `target = 9`

Output: `[0, 1]`

Explanation: Because `nums[0] + nums[1] == 9`, so return `[0, 1]`.

Input: `nums = [3, 2, 4]`, `target = 6`

Output: `[1, 2]`

Input: `nums = [3, 3]`, `target = 6`

Output: `[0, 1]`

Hints:

```
def twoSum(self, nums: List[int], target: int) -> List[int]:
    a=[]
    # Write code here
    ...

    return a
```

1.2 Contains Duplicate

Given an integer array `nums`, return `true` if any value appears at least twice in the array, and return `false` if every element is distinct.

Input: `nums = [1, 2, 3, 1]`

Output: `true`

Input: `nums = [1, 2, 3, 4]`

Output: `false`

Input: `nums = [1, 1, 1, 3, 3, 4, 3, 2, 4, 2]`

Output: `true`

Hints:

```
def containsDuplicate(self, nums):
    a = set() # set can have only distinct elements
    # Write code here
    ...

    return False
```

1.3 Roman to Integer

Roman numerals are represented by seven different symbols: I, V, X, L, C, D and M.

Symbol	Value
I	1
V	5
X	10
L	50
C	100
D	500
M	1000

For example, 2 is written as II in Roman numeral, just two ones added together. 12 is written as XII, which is simply X + II. The number 27 is written as XXVII, which is XX + V + II.

Roman numerals are usually written largest to smallest from left to right. However, the numeral for four is not IIII. Instead, the number four is written as IV. Because the one is before the five we subtract it making four. The same principle applies to the number nine, which is written as IX. There are six instances where subtraction is used:

I can be placed before V (5) and X (10) to make 4 and 9.

X can be placed before L (50) and C (100) to make 40 and 90.

C can be placed before D (500) and M (1000) to make 400 and 900.

Given a roman numeral, convert it to an integer.

Input: s = "III"

Output: 3

Input: s = "LVIII"

Output: 58

Hints:

```
def romanToInt(self, s: str) -> int:
    # Write code here
    ...

    return number
```

1.4 Plus One

You are given a large integer represented as an integer array digits, where each digits[i] is the ith digit of the integer. The digits are ordered from most significant to least significant in left-to-right order. The large integer does not contain any leading 0's. Increment the large integer by one and return the resulting array of digits.

Input: digits = [1, 2, 3]

Output: [1, 2, 4]

Explanation: The array represents the integer 123.

Incrementing by one gives 123 + 1 = 124.

Thus, the result should be [1, 2, 4].

Hints:

```
def plusOne(self, digits: List[int]) -> List[int]:
    n = len(digits)
    # Write code here
    ...

    return digits
```

1.5 Majority Element

Given an array nums of size n, return the majority element. The majority element is the element that appears more than $\lfloor n / 2 \rfloor$ times. You may assume that the majority element always exists in the array.

Input: nums = [3, 2, 3]

Output: 3

Input: nums = [2, 2, 1, 1, 1, 2, 2]

Output: 2

Hints:

```
def majorityElement(self, nums):
    # write code here
    ...
```

1.6 Richest Customer Wealth

You are given an $m \times n$ integer grid `accounts` where `accounts[i][j]` is the amount of money the i^{th} customer has in the j^{th} bank. Return the wealth that the richest customer has. A customer's wealth is the amount of money they have in all their bank accounts. The richest customer is the customer that has the maximum wealth.

Input: `accounts = [[1, 2, 3], [3, 2, 1]]`

Output: 6

Explanation:

1st customer has wealth = $1 + 2 + 3 = 6$

2nd customer has wealth = $3 + 2 + 1 = 6$

Both customers are considered the richest with a wealth of 6 each, so return 6.

Input: `accounts = [[1, 5], [7, 3], [3, 5]]`

Output: 10

Explanation:

1st customer has wealth = 6

2nd customer has wealth = 10

3rd customer has wealth = 8

The 2nd customer is the richest with a wealth of 10.

Input: `accounts = [[2, 8, 7], [7, 1, 3], [1, 9, 5]]`

Output: 17

Hints:

```
def maximumWealth(self, accounts: List[List[int]]) -> int:
```

```
    # write code here
```

```
    ...
```

1.7 Fizz Buzz

Given an integer n , return a string array `answer` (1-indexed) where:

`answer[i] == "FizzBuzz"` if i is divisible by 3 and 5.

`answer[i] == "Fizz"` if i is divisible by 3.

`answer[i] == "Buzz"` if i is divisible by 5.

`answer[i] == i` (as a string) if none of the above conditions are true.

Input: $n = 3$

Output: `["1", "2", "Fizz"]`

Input: $n = 5$

Output: `["1", "2", "Fizz", "4", "Buzz"]`

Input: $n = 15$

Output: `["1", "2", "Fizz", "4", "Buzz", "Fizz", "7", "8", "Fizz", "Buzz", "11", "Fizz", "13", "14", "FizzBuzz"]`

Hints:

```
def fizzBuzz(self, n: int) -> List[str]:
```

```
    # write code here
```

```
    ...
```

1.8 Number of Steps to Reduce a Number to Zero

Given an integer `num`, return the number of steps to reduce it to zero. In one step, if the current number is even, you have to divide it by 2, otherwise, you have to subtract 1 from it.

Input: `num = 14`

Output: 6

Explanation:

- 14 is even; divide by 2 and obtain 7.
- 7 is odd; subtract 1 and obtain 6.
- 6 is even; divide by 2 and obtain 3.
- 3 is odd; subtract 1 and obtain 2.
- 2 is even; divide by 2 and obtain 1.
- 1 is odd; subtract 1 and obtain 0.

Input: num = 8

Output: 4

Explanation:

- 8 is even; divide by 2 and obtain 4.
- 4 is even; divide by 2 and obtain 2.
- 2 is even; divide by 2 and obtain 1.
- 1 is odd; subtract 1 and obtain 0.

Input: num = 123

Output: 12

Hints:

```
def numberOfSteps(self, n: int) -> int:
    # write code here
    ...
```

1.9 Running Sum of 1D Array

Given an array nums. We define a running sum of an array as $\text{runningSum}[i] = \text{sum}(\text{nums}[0] \dots \text{nums}[i])$.

Return the running sum of nums.

Input: nums = [1, 2, 3, 4]

Output: [1, 3, 6, 10]

Explanation: Running sum is obtained as follows: [1, 1+2, 1+2+3, 1+2+3+4].

Input: nums = [1, 1, 1, 1, 1]

Output: [1, 2, 3, 4, 5]

Explanation: Running sum is obtained as follows: [1, 1+1, 1+1+1, 1+1+1+1, 1+1+1+1+1].

Input: nums = [3, 1, 2, 10, 1]

Output: [3, 4, 6, 16, 17]

Hints:

```
def runningSum(self, nums: List[int]) -> List[int]:
    # write code here
    ...
    return answer
```

1.10 Remove Element

Given an integer array nums and an integer val, remove all occurrences of val in nums in-place. The order of the elements may be changed. Then return the number of elements in nums which are not equal to val. Consider the number of elements in nums which are not equal to val be k, to get accepted, you need to do the following things:

- Change the array nums such that the first k elements of nums contain the elements which are not equal to val. The remaining elements of nums are not important as well as the size of nums.
- Return k.

Input: nums = [3, 2, 2, 3], val = 3

Output: 2, nums = [2, 2, _, _]

Explanation: Your function should return k = 2, with the first two elements of nums being 2.

It does not matter what you leave beyond the returned k (hence they are underscores).

Input: nums = [0,1,2,2,3,0,4,2], val = 2

Output: 5, nums = [0,1,4,0,3,_,_,_]

Explanation: Your function should return k = 5, with the first five elements of nums containing 0, 0, 1, 3, and 4.

Note that the five elements can be returned in any order.

It does not matter what you leave beyond the returned k (hence they are underscores).

Hints:

```
def removeElement(self, nums: List[int], val: int) -> int:
    # write code here
    ...
    return len(nums)
```

2. Matrix Operations

2.1 Add Two Matrices

Given two matrices X and Y, the task is to compute the sum of two matrices and then print it in Python.

Input:

```
X= [[1, 2, 3],
     [4, 5, 6],
     [7, 8, 9]]
```

```
Y = [[9, 8, 7],
     [6, 5, 4],
     [3, 2, 1]]
```

Output:

```
Result = [[10, 10, 10],
          [10, 10, 10],
          [10, 10, 10]]
```

Hints:

```
# Program to add two matrices using nested loop
```

```
X = [[1, 2, 3],
     [4, 5, 6],
     [7, 8, 9]]
```

```
Y = [[9,8,7],
     [6,5,4],
     [3,2,1]]
```

```
result = [[0,0,0],
          [0,0,0],
          [0,0,0]]
```

```
# iterate through rows
for i in range(len(X)):
```

```
    # write code here
```

```
    ...
```

```
for r in result:
```



```
print(r)
```

TRY

1. Take input as X = [[10, 20, 30],[41, 52, 63], [47, 58, 69]] Y = [[19,18,17],[66,35,49], [13,21,11]] and verify the results.

2.2 Multiply Two Matrices

Given two matrices X and Y, the task is to compute the multiplication of two matrices and then print it.

Input:

```
X= [[1, 7, 3],
     [3, 5, 6],
     [6, 8, 9]]
```

```
Y = [[1, 1, 1, 2],
     [6, 7, 3, 0],
     [4, 5, 9, 1]]
```

Output:

```
Result = [[55, 65, 49, 5],
          [57, 68, 72, 12],
          [90, 107, 111, 21]]
```

Hints:

```
# Program to multiply two matrices using list comprehension
```

```
# take a 3x3 matrix
```

```
A = [[12, 7, 3],
     [4, 5, 6],
     [7, 8, 9]]
```

```
# take a 3x4 matrix
```

```
B = [[5, 8, 1, 2],
     [6, 7, 3, 0],
     [4, 5, 9, 1]]
```

```
# result will be 3x4
```

```
# write code here
```

```
...
for r in result:
    print(r)
```

TRY

1. Take input as X = [[11, 0, 30],[-41, -2, 63], [41, -5, -9]] Y = [[19,-48,17],[-6,35,19], [13,1,-9]] and verify the results.

2.3 Transpose of a Matrix

A matrix can be implemented using a nested list. Each element is treated as a row of the matrix. Find the transpose of a matrix in multiple ways.

Input: [[1, 2], [3, 4], [5, 6]]

Output: [[1, 3, 5], [2, 4, 6]]

Explanation: Suppose we are given a matrix

```
[[1, 2],
 [3, 4],
 [5, 6]]
```

Then the transpose of the given matrix will be,

```
[[1, 3, 5],
 [2, 4, 6]]
```

Hints:

```
# Program to multiply two matrices using list comprehension
```

```
# take a 3x3 matrix
```

```
A = [[12, 7, 3],
      [4, 5, 6],
      [7, 8, 9]]
```

```
# result will be 3x4
```

```
# write code here
```

```
...
for r in result:
    print(r)
```

TRY

1. Take input as X = [[11, 0, 30], [-41, -2, 63], [41, -5, -9]] and verify the results.

2.4 Matrix Product

Matrix product problem we can solve using list comprehension as a potential shorthand to the conventional loops. Iterate and find the product of the nested list and at the end return the cumulative product using function.

Input: The original list: [[1, 4, 5], [7, 3], [4], [46, 7, 3]]

Output: The total element product in lists is: 1622880

Hints:

```
# Matrix Product using list comprehension + loop
```

```
def prod(val):
    # write code here
    ...
```

```
# initializing list
```

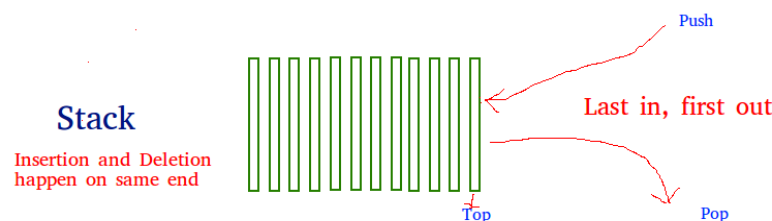
```
test_list = [[1, 4, 5], [7, 3], [4], [46, 7, 3]]
```

TRY

1. Take input list: [[1, 4, 5], [7, 3], [4], [46, 7, 3]] and verify the result.

3. Stack**3.1 Stack implementation using List**

A stack is a linear data structure that stores items in a Last-In/First-Out (LIFO) or First-In/Last-Out (FILO) manner. In stack, a new element is added at one end and an element is removed from that end only. The insert and delete operations are often called push and pop.



The functions associated with stack are:

- **empty()** – Returns whether the stack is empty
- **size()** – Returns the size of the stack
- **top() / peek()** – Returns a reference to the topmost element of the stack
- **push(a)** – Inserts the element 'a' at the top of the stack
- **pop()** – Deletes the topmost element of the stack

Hints:

```
# Stack implementation using list
```

```
top=0
```

```
mymax=5
```

```
def createStack():
```

```
    stack=[]
```

```
    return stack
```

```
def isEmpty(stack):
```

```
    # write code here
```

```
    ...
```

```
def Push(stack,item):
```

```
    # write code here
```

```
    ...
```

```
def Pop(stack):
```

```
    # write code here
```

```
    ...
```

```
# create a stack object
```

```
stack = createStack()
```

```
while True:
```

```
    print("1.Push")
```

```
    print("2.Pop")
```

```
    print("3.Display")
```

```
    print("4.Quit")
```

```
    # write code here
```

```
    ...
```

TRY

1. Take input operations as [PUSH(A),PUSH(B),PUSH(C),POP,POP,POP,POP,PUSH(D)] and verify the result.
2. Take input operations as [POP, POP, PUSH (A), PUSH (B), POP, and PUSH(C)] and verify the result.

3.2 Balanced Parenthesis Checking

Given an expression string, write a python program to find whether a given string has balanced parentheses or not.

Input: {[]{}() }

Output: Balanced

Input: [{ } { } (]

Output: Unbalanced

Using stack One approach to check balanced parentheses is to use stack. Each time, when an open parentheses is encountered push it in the stack, and when closed parenthesis is encountered, match it with the top of stack and pop it. If stack is empty at the end, return Balanced otherwise, Unbalanced.

Hints:

```
# Check for balanced parentheses in an expression
```

```
open_list = ["[","{","("]
```

```
close_list = ["]","}",")"]
```

```
# Function to check parentheses
```

```
def check(myStr):
```

```
    # write code here    ...
```

TRY

1. Take input as $\{\{\}\{\}\{\}\{\}\}$ and verify the result.
2. Take input as $\{\{\}\{\}\{\}\{\}\{\}\}$ and verify the result.

3.3 Evaluation of Postfix Expression

Given a postfix expression, the task is to evaluate the postfix expression. Postfix expression: The expression of the form “a b operator” (ab+) i.e., when a pair of operands is followed by an operator.

Input: str = “2 3 1 * + 9 -“

Output: -4

Explanation: If the expression is converted into an infix expression, it will be $2 + (3 * 1) - 9 = 5 - 9 = -4$.

Input: str = “100 200 + 2 / 5 * 7 +”

Output: 757

Procedure for evaluation postfix expression using stack:

- Create a stack to store operands (or values).
- Scan the given expression from left to right and do the following for every scanned element.
 - If the element is a number, push it into the stack.
 - If the element is an operator, pop operands for the operator from the stack. Evaluate the operator and push the result back to the stack.
- When the expression is ended, the number in the stack is the final answer.

Hints:

Evaluate value of a postfix expression

Class to convert the expression

class Evaluate:

Constructor to initialize the class variables

def __init__(self, capacity):

self.top = -1

self.capacity = capacity

This array is used as a stack

self.array = []

Check if the stack is empty

def isEmpty(self):

write code here

...

def peek(self):

write code here

...

def pop(self):

write code here

...

def push(self, op):

write code here

...

def evaluatePostfix(self, exp):

write code here

...

```
# Driver code
exp = "231*+9-"
obj = Evaluate(len(exp))

# Function call
print("postfix evaluation: %d" % (obj.evaluatePostfix(exp)))
```

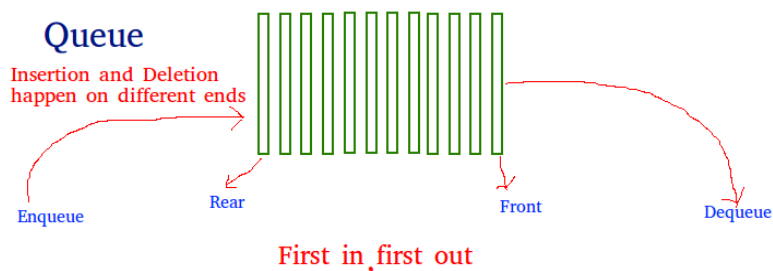
TRY

1. Take input str = "A B + C* D / E +" and verify the result.
2. Take input str = "XYZ- + W+ R / S -" and verify the result.

4. Queue

4.1 Linear Queue using List

Linear queue is a linear data structure that stores items in First in First out (FIFO) manner. With a queue the least recently added item is removed first. A good example of queue is any queue of consumers for a resource where the consumer that came first is served first.



Hints:

```
# Static implementation of linear queue
front=0
rear=0
mymax=5
def createQueue():
    queue=[] #empty list
    return queue

def isEmpty(queue):
    # write code here
    ...

def enqueue(queue,item): # insert an element into the queue
    # write code here
    ...

def dequeue(queue): #remove an element from the queue
    # write code here
    ...

# Driver code
queue = createQueue()
while True:
    print("1.Enqueue")
    print("2.Dequeue")
    print("3.Display")
```

```
print("4.Quit")
# write code here
...
```

TRY

1. Take input operations as [ENQUEUE(A),DEQUEUE(),ENQUEUE(B),DEQUEUE(),ENQUEUE(C),DEQUEUE(),] and verify the result.
2. Take input operations as [ENQUEUE(A), ENQUEUE(B),DEQUEUE(),ENQUEUE(C), DEQUEUE(),ENQUEUE(D), DEQUEUE(), ENQUEUE(C),DEQUEUE(),] and verify the result.

4.2 Stack using Queues

Implement a last-in-first-out (LIFO) stack using only two queues. The implemented stack should support all the functions of a normal stack (push, top, pop, and empty).

- void push(int x) Pushes element x to the top of the stack.
- int pop() Removes the element on the top of the stack and returns it.
- int top() Returns the element on the top of the stack.
- boolean empty() Returns true if the stack is empty, false otherwise.

Input:

```
["MyStack", "push", "push", "top", "pop", "empty"]
[[], [1], [2], [], [], []]
```

Output:

```
[null, null, null, 2, 2, false]
```

Hints:

```
class MyStack:

    def __init__(self):
        # write code here
        ...

    def push(self, x: int) -> None:
        # write code here
        ...

    def pop(self) -> int:
        # write code here
        ...

    def top(self) -> int:
        # write code here
        ...

    def empty(self) -> bool:
        # write code here
        ...

# Your MyStack object will be instantiated and called as such:
# obj = MyStack()
# obj.push(x)
# param_2 = obj.pop()
# param_3 = obj.top()
# param_4 = obj.empty()
```

4.3 Implement Queue using Stacks

Implement a first in first out (FIFO) queue using only two stacks. The implemented queue should support all the functions of a normal queue (push, peek, pop, and empty).

- void push(int x) Pushes element x to the back of the queue.
- int pop() Removes the element from the front of the queue and returns it.
- int peek() Returns the element at the front of the queue.
- boolean empty() Returns true if the queue is empty, false otherwise.

Input:

["MyQueue", "push", "push", "peek", "pop", "empty"]

[[], [1], [2], [], [], []]

Output:

[null, null, null, 1, 1, false]

Hints:

```
class MyQueue:

    def __init__(self):
        # write code here
        ...

    def push(self, x: int) -> None:
        # write code here
        ...

    def pop(self) -> int:
        # write code here
        ...

    def peek(self) -> int:
        # write code here
        ...

    def empty(self) -> bool:
        # write code here
        ...

# Your MyQueue object will be instantiated and called as such:
# obj = MyQueue()
# obj.push(x)
# param_2 = obj.pop()
# param_3 = obj.peek()
# param_4 = obj.empty()
```

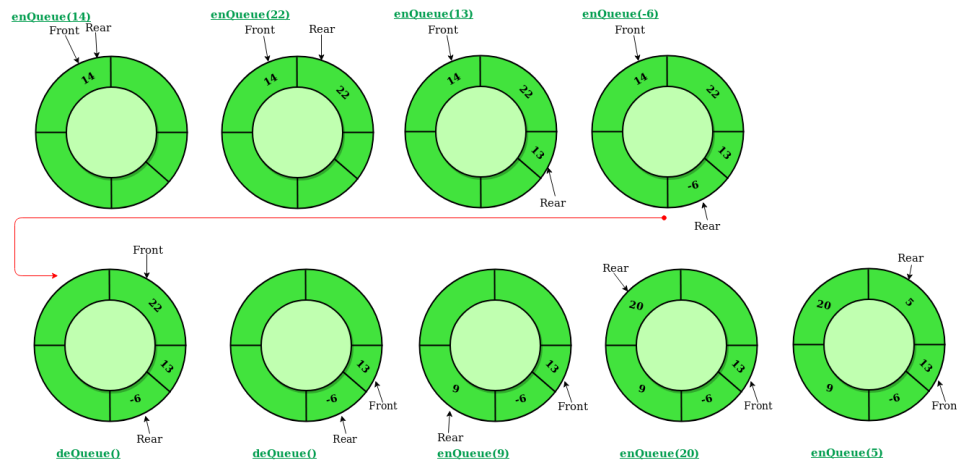
4.4 Circular Queue

A Circular Queue is an extended version of a normal queue where the last element of the queue is connected to the first element of the queue forming a circle. The operations are performed based on FIFO (First In First Out) principle. It is also called 'Ring Buffer'.

Operations on Circular Queue:

- **Front:** Get the front item from the queue.
- **Rear:** Get the last item from the queue.
- **enQueue(value)** This function is used to insert an element into the circular queue. In a circular queue, the new element is always inserted at the rear position.

- Check whether the queue is full – [i.e., the rear end is in just before the front end in a circular manner].
- If it is full then display Queue is full.
- If the queue is not full then, insert an element at the end of the queue.
- **deQueue()** This function is used to delete an element from the circular queue. In a circular queue, the element is always deleted from the front position.
- Check whether the queue is Empty.
- If it is empty then display Queue is empty.
- If the queue is not empty, then get the last element and remove it from the queue.



Implement Circular Queue using Array:

1. Initialize an array queue of size **n**, where **n** is the maximum number of elements that the queue can hold.
2. Initialize two variables front and rear to -1.
3. **Enqueue:** To enqueue an element **x** into the queue, do the following:
 - Increment rear by 1.
 - If **rear** is equal to **n**, set **rear** to 0.
 - If **front** is -1, set **front** to 0.
 - Set queue[rear] to **x**.
4. **Dequeue:** To dequeue an element from the queue, do the following:
 - Check if the queue is empty by checking if **front** is -1.
 - If it is, return an error message indicating that the queue is empty.
 - Set **x** to queue [front].
 - If **front** is equal to **rear**, set **front** and **rear** to -1.
 - Otherwise, increment **front** by 1 and if **front** is equal to **n**, set **front** to 0.
 - Return **x**.

Hints:

```
class CircularQueue():

    # constructor
    def __init__(self, size): # initializing the class
        self.size = size

    # initializing queue with none
    self.queue = [None for i in range(size)]
    self.front = self.rear = -1

    def enqueue(self, data):
        # Write code here
        ...

    def dequeue(self):
        # Write code here
        ...
```



```

def display(self):
    # Write code here
    ...

# Driver Code
ob = CircularQueue(5)
ob.enqueue(14)
ob.enqueue(22)
ob.enqueue(13)
ob.enqueue(-6)
ob.display()
print ("Deleted value = ", ob.dequeue())
print ("Deleted value = ", ob.dequeue())
ob.display()
ob.enqueue(9)
ob.enqueue(20)
ob.enqueue(5)
ob.display()

```

TRY

1. Take input operations as [ENQUEUE(A,B,C,D,E,F),DEQUEUE(),DEQUEUE(),DEQUEUE(),ENQUEUE(G,H,I)] and verify the result.
2. Take input operations as [DEQUEUE(),ENQUEUE(A,B,C,D,E,F),DEQUEUE(),ENQUEUE(G,H,I)] and verify the result.

5. Graph Representation

5.1 Build a graph

You are given an integer n . Determine if there is an unconnected graph with n vertices that contains at least two connected components and contains the number of edges that is equal to the number of vertices. Each vertex must follow one of these conditions:

- Its degree is less than or equal to 1.
- It's a cut-vertex.

Note:

- The graph must be simple.
- Loops and multiple edges are not allowed.

Input: First line: n

Output: Print Yes if it is an unconnected graph. Otherwise, print No.

Sample Input	Sample Output
3	No

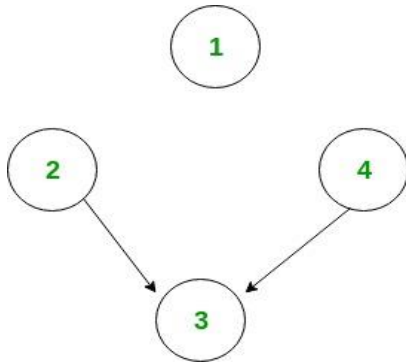
Constraints: $1 \leq n \leq 100$

Explanation: There is only one graph with the number of edges equal to the number of vertices (triangle) which is connected.

5.2 Number of Sink Nodes in a Graph

Given a Directed Acyclic Graph of n nodes (numbered from 1 to n) and m edges. The task is to find the number of sink nodes. A sink node is a node such that no edge emerges out of it.

Input: $n = 4$, $m = 2$, edges[] = {{2, 3}, {4, 3}}



Only node 1 and node 3 are sink nodes.

Input: $n = 4$, $m = 2$, $\text{edges[]} = \{\{3, 2\}, \{3, 4\}\}$

Output: 3

The idea is to iterate through all the edges. And for each edge, mark the source node from which the edge emerged out. Now, for each node check if it is marked or not. And count the unmarked nodes.

Algorithm:

1. Make any array $A[]$ of size equal to the number of nodes and initialize to 1.
2. Traverse all the edges one by one, say, $u \rightarrow v$.
 - (i) Mark $A[u]$ as 1.
3. Now traverse whole array $A[]$ and count number of unmarked nodes.

Hints:

```

# Program to count number of sink nodes

# Return the number of Sink Nodes.
def countSink(n, m, edgeFrom, edgeTo):
    # Write code here
    ...

    return count

# Driver Code
n = 4
m = 2
edgeFrom = [2, 4]
edgeTo = [3, 3]

print(countSink(n, m, edgeFrom, edgeTo))
  
```

5.3 Connected Components in a Graph

Given n , i.e. total number of nodes in an undirected graph numbered from 1 to n and an integer e , i.e. total number of edges in the graph. Calculate the total number of connected components in the graph. A connected component is a set of vertices in a graph that are linked to each other by paths.

Input: First line of input line contains two integers' n and e . Next e line will contain two integers u and v meaning that node u and node v are connected to each other in undirected fashion.

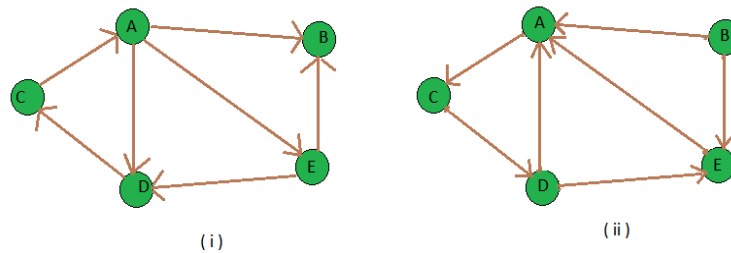
Output: For each input graph print an integer x denoting total number of connected components.

Sample Input	Sample Output
8 5	3
1 2	
2 3	
2 4	
3 5	
6 7	

Constraints: All the input values are well within the integer range.

5.4 Transpose Graph

Transpose of a directed graph G is another directed graph on the same set of vertices with all of the edges reversed compared to the orientation of the corresponding edges in G . That is, if G contains an edge (u, v) then the converse/transpose/reverse of G contains an edge (v, u) and vice versa. Given a graph (represented as adjacency list), we need to find another graph which is the transpose of the given graph.



Input: figure (i) is the input graph.

Output: figure (ii) is the transpose graph of the given graph.

```
0--> 2
1--> 0 4
2--> 3
3--> 0 4
4--> 0
```

Explanation: We traverse the adjacency list and as we find a vertex v in the adjacency list of vertex u which indicates an edge from u to v in main graph, we just add an edge from v to u in the transpose graph i.e. add u in the adjacency list of vertex v of the new graph. Thus traversing lists of all vertices of main graph we can get the transpose graph.

Hints:

```
# find transpose of a graph.
# function to add an edge from vertex source to vertex dest
def addEdge(adj, src, dest):
    adj[src].append(dest)

# function to print adjacency list of a graph
def displayGraph(adj, v):
    ...
    print()

# function to get Transpose of a graph taking adjacency list of given graph and that of Transpose graph
def transposeGraph(adj, transpose, v):
    # traverse the adjacency list of given graph and for each edge (u, v) add
    # an edge (v, u) in the transpose graph's adjacency list
    ...

# Driver Code
v = 5
adj = [[] for i in range(v)]
addEdge(adj, 0, 1)
addEdge(adj, 0, 4)
addEdge(adj, 0, 3)
addEdge(adj, 2, 0)
addEdge(adj, 3, 2)
addEdge(adj, 4, 1)
addEdge(adj, 4, 3)

# Finding transpose of graph represented by adjacency list adj[]
transpose = [[] for i in range(v)]
transposeGraph(adj, transpose, v)

# Displaying adjacency list of transpose graph i.e. b
```

TRY

1. Take input operations as addEdge(A, B), addEdge(A, D), addEdge(A, C), addEdge(C, A), addEdge(A, D), addEdge(C, B), addEdge(B, C) and verify the result.

5.5 Counting Triplets

You are given an undirected, complete graph G that contains N vertices. Each edge is colored in either white or black. You are required to determine the number of triplets (i, j, k) ($1 \leq i < j < k \leq N$) of vertices such that the edges (i, j) , (j, k) , (i, k) are of the same color.

There are M white edges and $(N(N-1)/2) - M$ black edges.

Input:

First line: Two integers – N and M ($3 \leq N \leq 10^5$, $1 \leq M \leq 3 * 10^5$)

$(i+1)^{\text{th}}$ line: Two integers – u_i and v_i ($1 \leq u_i, v_i \leq N$) denoting that the edge (u_i, v_i) is white in color.

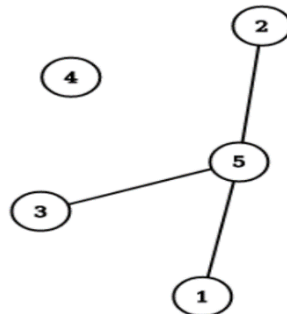
Note: The conditions $(u_i, v_i) \neq (u_j, v_j)$ and $(u_i, v_i) \neq (v_j, u_j)$ are satisfied for all $1 \leq i < j \leq M$.

Output: Print an integer that denotes the number of triples that satisfy the mentioned condition.

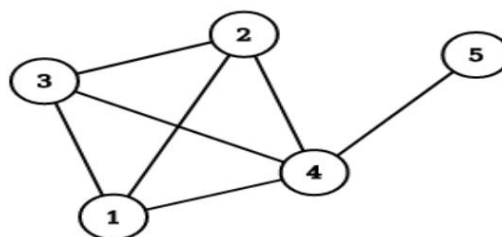
Sample Input	Sample Output
5 3	4
1 5	
2 5	
3 5	

Explanation: The triplets are: $\{(1, 2, 3), (1, 2, 4), (2, 3, 4), (1, 3, 4)\}$

The graph consisting of only white edges:



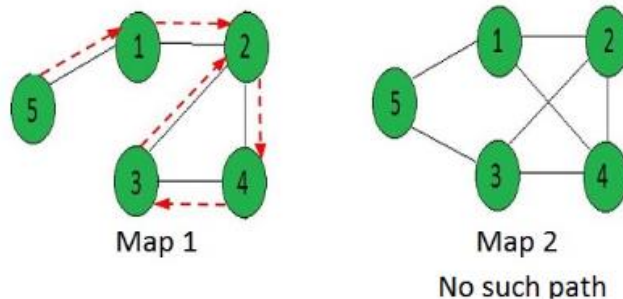
The graph consisting of only black edges:

**6. Graph Routing Algorithms**

6.1 Seven Bridges of Königsberg

There was 7 bridges connecting 4 lands around the city of Königsberg in Prussia. Was there any way to start from any of the land and go through each of the bridges once and only once? Euler first introduced graph theory to solve this problem. He considered each of the lands as a node of a graph and each bridge in between as an edge in between. Now he calculated if there is any Eulerian Path in that graph. If there is an Eulerian path then there is a solution otherwise not.

There are n nodes and m bridges in between these nodes. Print the possible path through each node using each edges (if possible), traveling through each edges only once.



Input: [[0, 1, 0, 0, 1],
[1, 0, 1, 1, 0],
[0, 1, 0, 1, 0],
[0, 1, 1, 0, 0],
[1, 0, 0, 0, 0]]

Output: 5 -> 1 -> 2 -> 4 -> 3 -> 2

Input: [[0, 1, 0, 1, 1],
[1, 0, 1, 0, 1],
[0, 1, 0, 1, 1],
[1, 1, 1, 0, 0],
[1, 0, 1, 0, 0]]

Output: "No Solution"

Hints:

```
# A Python program to print Eulerian trail in a
# given Eulerian or Semi-Eulerian Graph
from collections import defaultdict

class Graph:
    # Constructor and destructor
    def __init__(self, V):
        self.V = V
        self.adj = defaultdict(list)

    # functions to add and remove edge
```

```

def addEdge(self, u, v):
def rmvEdge(self, u, v):
    ...
# Methods to print Eulerian tour
def printEulerTour(self):
    # Find a vertex with odd degree
    ...
    # Print tour starting from oddv
self.printEulerUtil(u)
print()
def printEulerUtil(self, u):
    # Recur for all the vertices adjacent to this vertex
for v in self.adj[u]:
    # If edge u-v is not removed and it's a valid next edge
    # The function to check if edge u-v can be considered as next edge in Euler Tout
    def isValidNextEdge(self, u, v):
        # The edge u-v is valid in one of the following two cases:

        # 1) If v is the only adjacent vertex of u
        ...

        # 2) If there are multiple adjacents, then u-v is not a bridge
        # Do following steps to check if u-v is a bridge
        # 2.a) count of vertices reachable from u
        ...

        # 2.b) Remove edge (u, v) and after removing
        # the edge, count vertices reachable from u
        ...

        # 2.c) Add the edge back to the graph
        self.addEdge(u, v)

        # 2.d) If count1 is greater, then edge (u, v) is a bridge
        return False if count1 > count2 else True

    # A DFS based function to count reachable vertices from v

    def DFSCount(self, v, visited):
        # Mark the current node as visited
        ...
        # Recur for all the vertices adjacent to this vertex
        ...

    # utility function to form edge between two vertices source and dest
    def makeEdge(src, dest):
        graph.addEdge(src, dest)

# Driver code
# Let us first create and test graphs shown in above figure
g1 = Graph(4)
g1.addEdge(0, 1)
g1.addEdge(0, 2)
g1.addEdge(1, 2)
g1.addEdge(2, 3)
g1.printEulerTour()

g3 = Graph(4)
g3.addEdge(0, 1)

```

```

g3.addEdge(1, 0)
g3.addEdge(0, 2)
g3.addEdge(2, 0)
g3.addEdge(2, 3)
g3.addEdge(3, 1)
g3.printEulerTour()

```

TRY

1. Take input: `[[1, 0, 1, 0, 1], [1, 0, 1, 0, 0], [1, 1, 0, 1, 0], [0, 0, 1, 0, 0], [1, 0, 1, 0, 0]]` and verify the result.
2. Take input: `[[0, 0, 1, 0, 1], [0, 0, 1, 0, 0], [1, 0, 0, 1, 0], [1, 0, 1, 0, 0], [1, 1, 1, 0, 0]]` and verify the result.

6.2 Hamiltonian Cycle

The Hamiltonian cycle of undirected graph $G = \langle V, E \rangle$ is the cycle containing each vertex in V . If graph contains a Hamiltonian cycle, it is called Hamiltonian graph otherwise it is non-Hamiltonian.

Hamiltonian Path in an undirected graph is a path that visits each vertex exactly once. A Hamiltonian cycle (or Hamiltonian circuit) is a Hamiltonian path such that there is an edge (in the graph) from the last vertex to the first vertex of the Hamiltonian path. Consider a graph G , and determine whether a given graph contains Hamiltonian cycle or not. If it contains, then prints the path. Following are the input and output of the required function.

Input: A 2D array graph $[V][V]$ where V is the number of vertices in graph and graph $[V][V]$ is adjacency matrix representation of the graph. A value graph $[i][j]$ is 1 if there is a direct edge from i to j , otherwise graph $[i][j]$ is 0.

Output: An array path $[V]$ that should contain the Hamiltonian Path. Path $[i]$ should represent the i^{th} vertex in the Hamiltonian Path. The code should also return false if there is no Hamiltonian Cycle in the graph.

For example, a Hamiltonian Cycle in the following graph is `{0, 1, 2, 4, 3, 0}`.

`(0)--(1)--(2)`

```

|  /\  |
|  /\  |
|  /\  |

```

`(3)-----(4)`

And the following graph doesn't contain any Hamiltonian Cycle.

`(0)--(1)--(2)`

```

|  /\  |
|  /\  |
|  /\  |

```

`(3) (4)`

Backtracking Algorithm: Create an empty path array and add vertex 0 to it. Add other vertices, starting from the vertex 1. Before adding a vertex, check for whether it is adjacent to the previously added vertex and not already added. If we find such a vertex, we add the vertex as part of the solution. If we do not find a vertex then we return false.

Hints:

```
# Python program for solution of Hamiltonian cycle problem
```

```

class Graph():
    def __init__(self, vertices):
        self.graph = [[0 for column in range(vertices)]
                      for row in range(vertices)]
        self.V = vertices

    def isSafe(self, v, pos, path):
        # Check if current vertex and last vertex in path are adjacent
        ...

    # A recursive utility function to solve Hamiltonian cycle problem
    def hamCycleUtil(self, path, pos):
        ...

    def hamCycle(self):
        ...

    def printSolution(self, path):
        print ("Solution Exists: Following", "is one Hamiltonian Cycle")
        for vertex in path:
            print (vertex, end = " ")
        print (path[0], "\n")

# Driver Code

''' Let us create the following graph
(0)--(1)--(2)
 |  /\  |
 |  /\  |
 |  /\  |
(3)-----(4) '''
g1 = Graph(5)
g1.graph = [ [0, 1, 0, 1, 0], [1, 0, 1, 1, 1],
             [0, 1, 0, 0, 1], [1, 1, 0, 0, 1],
             [0, 1, 1, 1, 0], ]

# Print the solution
g1.hamCycle();

''' Let us create the following graph
(0)--(1)--(2)
 |  /\  |
 |  /\  |
 |  /\  |
(3)   (4) '''
g2 = Graph(5)
g2.graph = [ [0, 1, 0, 1, 0], [1, 0, 1, 1, 1],
             [0, 1, 0, 0, 1], [1, 1, 0, 0, 0],
             [0, 1, 1, 0, 0], ]

# Print the solution
g2.hamCycle();

```

TRY

1. Take a graph = [[1, 1, 0, 1, 0], [1, 1, 1, 1, 1], [0, 1, 0, 1, 1], [1, 1, 0, 1, 0], [0, 1, 1, 1, 0],] and verify the results.

6.3 Number of Hamiltonian Cycle

Given an undirected complete graph of N vertices where $N > 2$. The task is to find the number of different Hamiltonian cycle of the graph.

Complete Graph: A graph is said to be complete if each possible vertices is connected through an Edge.

Hamiltonian Cycle: It is a closed walk such that each vertex is visited at most once except the initial vertex. and it is not necessary to visit all the edges.

Formula: $(N - 1)! / 2$

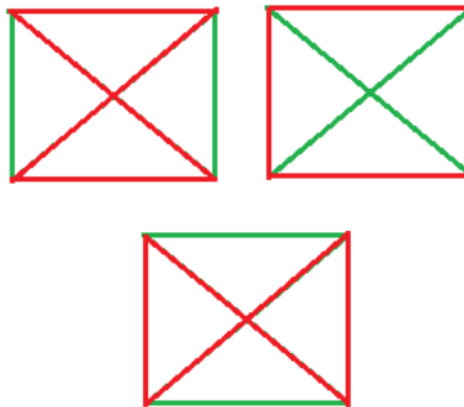
Input: $N = 6$

Output: Hamiltonian cycles = 60

Input: $N = 4$

Output: Hamiltonian cycles = 3

Explanation: Let us take the example of $N = 4$ complete undirected graph, The 3 different Hamiltonian cycle is as shown below:



Hints:

```
# Number of Hamiltonian cycles
import math as mt

# Function that calculates number of Hamiltonian cycle
def Cycles(N):
    ...

# Driver code
N = 5
Number = Cycles(N)
print("Hamiltonian cycles = ", Number)
```

TRY

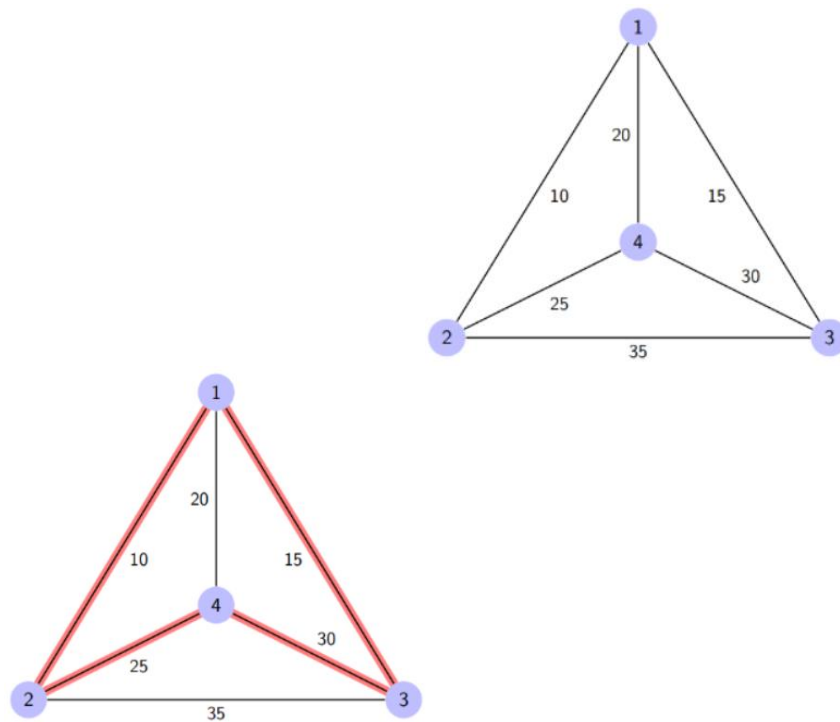
1. Take an input $N=7$ and verify the results.
2. Take an input $N=10$ and verify the results.

7. Shortest Path Algorithms

7.1 Travelling Salesman Problem

Given a set of cities and the distance between every pair of cities, the problem is to find the shortest possible route that visits every city exactly once and returns to the starting point. The problem statement gives a list of cities along with the distances between each city.

Objective: To start from the origin city, visit other cities only once, and return to the original city again. Our target is to find the shortest possible path to complete the round-trip route.



Here a graph is given where 1, 2, 3, and 4 represent the cities, and the weight associated with every edge represents the distance between those cities. The goal is to find the shortest possible path for the tour that starts from the origin city, traverses the graph while only visiting the other cities or nodes once, and returns to the origin city.

For the above graph, the optimal route is to follow the minimum cost path: 1 – 2 – 4 – 3 – 1. And this shortest route would cost $10 + 25 + 30 + 15 = 80$

Algorithm for Traveling Salesman Problem: We will use the dynamic programming approach to solve the Travelling Salesman Problem (TSP).

- A graph $G=(V, E)$, which is a set of vertices and edges.
- V is the set of vertices.
- E is the set of edges.
- Vertices are connected through edges.
- $\text{Dist}(i,j)$ denotes the non-negative distance between two vertices, i and j .

Let's assume S is the subset of cities and belongs to $\{1, 2, 3, \dots, n\}$ where $1, 2, 3 \dots n$ are the cities and i, j are two cities in that subset. Now $\text{cost}(i, S, j)$ is defined in such a way as the length of the shortest path visiting node in S , which is exactly once having the starting and ending point as i and j respectively.

For example, $\text{cost}(1, \{2, 3, 4\}, 1)$ denotes the length of the shortest path where:

- Starting city is 1
- Cities 2, 3, and 4 are visited only once
- The ending point is 1

The dynamic programming algorithm would be:

- Set $\text{cost}(i, i) = 0$, which means we start and end at i , and the cost is 0.
- When $|S| > 1$, we define $\text{cost}(i, S, 1) = \infty$ where $i \neq 1$. Because initially, we do not know the exact cost to reach city i to city 1 through other cities.
- Now, we need to start at 1 and complete the tour. We need to select the next city in such a way-
- $\text{cost}(i, S, j) = \min \text{cost}(i, S - \{i\}, j) + \text{dist}(i, j)$ where $i \in S$ and $i \neq j$

For the given figure above, the adjacency matrix would be the following:

dist(i, j)	1	2	3	4
1	0	10	15	20
2	10	0	35	25
3	15	35	0	30
4	20	25	30	0

Now $S = \{1, 2, 3, 4\}$. There are four elements. Hence the number of subsets will be 2^4 or 16. Those subsets are-

1) $|S| = \text{Null}$:

$\{\Phi\}$

2) $|S| = 1$:

$\{\{1\}, \{2\}, \{3\}, \{4\}\}$

3) $|S| = 2$:

$\{\{1, 2\}, \{1, 3\}, \{1, 4\}, \{2, 3\}, \{2, 4\}, \{3, 4\}\}$

4) $|S| = 3$:

$\{\{1, 2, 3\}, \{1, 2, 4\}, \{2, 3, 4\}, \{1, 3, 4\}\}$

5) $|S| = 4$:

$\{\{1, 2, 3, 4\}\}$

As we are starting at 1, we could discard the subsets containing city 1. The algorithm calculation steps:

1) $|S| = \Phi$:

$\text{cost}(2, \Phi, 1) = \text{dist}(2, 1) = 10$

$\text{cost}(3, \Phi, 1) = \text{dist}(3, 1) = 15$

$\text{cost}(4, \Phi, 1) = \text{dist}(4, 1) = 20$

2) $|S| = 1$:

$\text{cost}(2, \{3\}, 1) = \text{dist}(2, 3) + \text{cost}(3, \Phi, 1) = 35 + 15 = 50$

$\text{cost}(2, \{4\}, 1) = \text{dist}(2, 4) + \text{cost}(4, \Phi, 1) = 25 + 20 = 45$

$\text{cost}(3, \{2\}, 1) = \text{dist}(3, 2) + \text{cost}(2, \Phi, 1) = 35 + 10 = 45$

$\text{cost}(3, \{4\}, 1) = \text{dist}(3, 4) + \text{cost}(4, \Phi, 1) = 30 + 20 = 50$

$\text{cost}(4, \{2\}, 1) = \text{dist}(4, 2) + \text{cost}(2, \Phi, 1) = 25 + 10 = 35$

$\text{cost}(4, \{3\}, 1) = \text{dist}(4, 3) + \text{cost}(3, \Phi, 1) = 30 + 15 = 45$

3) $|S| = 2$:

$\text{cost}(2, \{3, 4\}, 1) = \min [\text{dist}[2,3] + \text{Cost}(3,\{4\},1) = 35+50 = 85,$

$\text{dist}[2,4] + \text{Cost}(4,\{3\},1) = 25+45 = 70] = 70$

$\text{cost}(3, \{2, 4\}, 1) = \min [\text{dist}[3,2] + \text{Cost}(2,\{4\},1) = 35+45 = 80,$

$\text{dist}[3,4] + \text{Cost}(4,\{2\},1) = 30+35 = 65] = 65$

$\text{cost}(4, \{2, 3\}, 1) = \min [\text{dist}[4,2] + \text{Cost}(2,\{3\},1) = 25+50 = 75$

$\text{dist}[4,3] + \text{Cost}(3,\{2\},1) = 30+45 = 75] = 75$

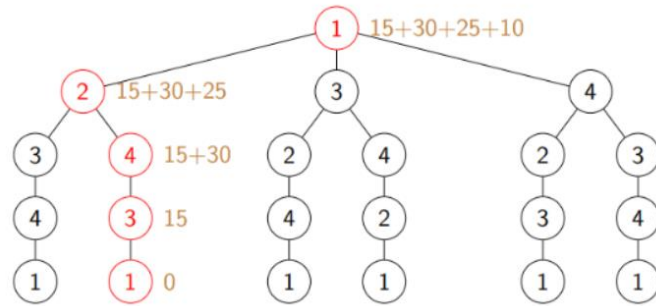
4) $|S| = 3$:

$\text{cost}(1, \{2, 3, 4\}, 1) = \min [\text{dist}[1,2] + \text{Cost}(2,\{3,4\},1) = 10+70 = 80$

$\text{dist}[1,3] + \text{Cost}(3,\{2,4\},1) = 15+65 = 80$

$\text{dist}[1,4] + \text{Cost}(4,\{2,3\},1) = 20+75 = 95] = 80$

So the optimal solution would be 1-2-4-3-1



Hints:

```
from sys import maxsize
from itertools import permutations
V = 4
def tsp(graph, s):
    ...
```

Driver code

```
graph = [[0, 10, 15, 20], [10, 0, 35, 25], [15, 35, 0, 30], [20, 25, 30, 0]]
s = 0
print(tsp(graph, s))
```

TRY

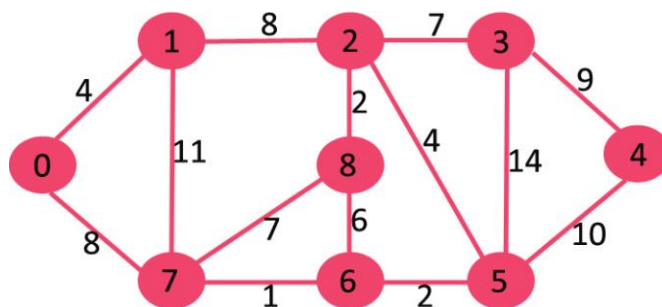
1. Take a below table values and verify the results.

dist(i, j)	1	2	3	4
1	0	40	25	40
2	20	0	35	25
3	25	35	0	60
4	40	25	30	0

7.2 Shortest Paths from Source to all Vertices (Dijkstra's Algorithm)

Given a graph and a source vertex in the graph, find the shortest paths from the source to all vertices in the given graph.

Input: src = 0, the graph is shown below.



Output: 0 4 12 19 21 11 9 8 14

Explanation: The distance from 0 to 1 = 4.
The minimum distance from 0 to 2 = 12. 0->1->2
The minimum distance from 0 to 3 = 19. 0->1->2->3

The minimum distance from 0 to 4 = 21. 0->7->6->5->4
The minimum distance from 0 to 5 = 11. 0->7->6->5
The minimum distance from 0 to 6 = 9. 0->7->6
The minimum distance from 0 to 7 = 8. 0->7
The minimum distance from 0 to 8 = 14. 0->1->2->8

Hints:

Dijkstra's single source shortest path algorithm. The program is for adjacency matrix representation of the graph

Library for INT_MAX

import sys

class Graph():

def __init__(self, vertices):

self.V = vertices

self.graph = [[0 for column in range(vertices)]
for row in range(vertices)]

def printSolution(self, dist):

print("Vertex \tDistance from Source")

for node in range(self.V):

print(node, "\t", dist[node])

A utility function to find the vertex with minimum distance value,

from the set of vertices not yet included in shortest path tree

def minDistance(self, dist, sptSet):

...

Function that implements Dijkstra's single source shortest path

algorithm for a graph represented using adjacency matrix representation

def dijkstra(self, src):

...

Driver's code

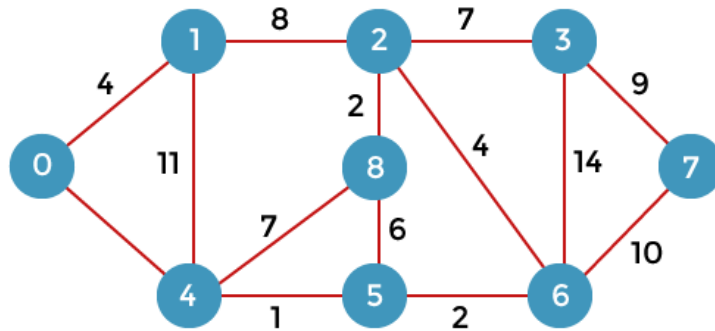
g = Graph(9)

g.graph = [[0, 4, 0, 0, 0, 0, 0, 8, 0],
[4, 0, 8, 0, 0, 0, 0, 11, 0],
[0, 8, 0, 7, 0, 4, 0, 0, 2],
[0, 0, 7, 0, 9, 14, 0, 0, 0],
[0, 0, 0, 9, 0, 10, 0, 0, 0],
[0, 0, 4, 14, 10, 0, 2, 0, 0],
[0, 0, 0, 0, 0, 2, 0, 1, 6],
[8, 11, 0, 0, 0, 0, 1, 0, 7],
[0, 0, 2, 0, 0, 0, 6, 7, 0]
]

g.dijkstra(0)

TRY

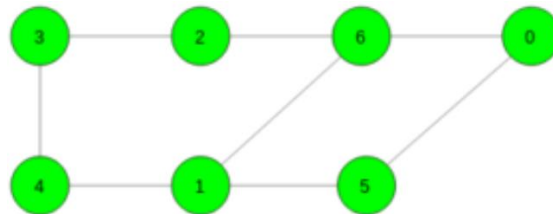
1. Take a below graph and verify the results.



7.3 Shortest Cycle in an Undirected Unweighted Graph

Given an undirected unweighted graph. The task is to find the length of the shortest cycle in the given graph. If no cycle exists print -1.

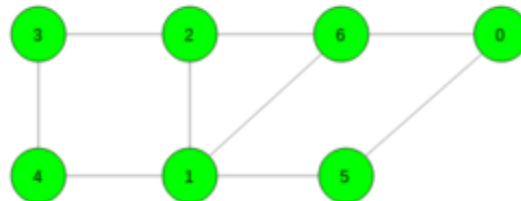
Input: Consider the graph given below



Output: 4

Cycle 6 -> 1 -> 5 -> 0 -> 6

Input: Consider the graph given below



Output: 3

Cycle 6 -> 1 -> 2 -> 6

Hints:

```
from sys import maxsize as INT_MAX
from collections import deque
```

```
N = 100200
```

```
gr = [0] * N
for i in range(N):
    gr[i] = []
```

```
# Function to add edge
```

```
def add_edge(x: int, y: int) -> None:
    global gr
```

```

gr[x].append(y)
gr[y].append(x)

# Function to find the length of the shortest cycle in the graph
def shortest_cycle(n: int) -> int:

    # To store length of the shortest cycle
    ans = INT_MAX

    # For all vertices
    # write code here
    ...

    # If graph contains no cycle
    if ans == INT_MAX:
        return -1

    # If graph contains cycle
    else:
        return ans

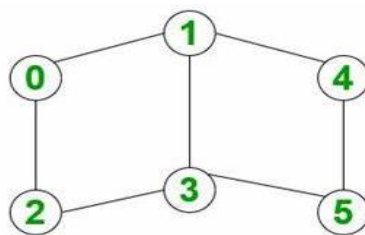
# Driver Code
# Number of vertices
n = 7
# Add edges
add_edge(0, 6)
add_edge(0, 5)
add_edge(5, 1)
add_edge(1, 6)
add_edge(2, 6)
add_edge(2, 3)
add_edge(3, 4)
add_edge(4, 1)

# Function call
print(shortest_cycle(n))

```

TRY

1. Take a below graph and verify the results.



7.4 Count Unique and all Possible Paths in a M x N Matrix

Count unique paths: The problem is to count all unique possible paths from the top left to the bottom right of a M X N matrix with the constraints that from each cell you can either move only to the right or down.

Input: M = 2, N = 2

Output: 2

Explanation: There are two paths

(0, 0) -> (0, 1) -> (1, 1)

(0, 0) -> (1, 0) -> (1, 1)

Input: M = 2, N = 3

Output: 3

Explanation: There are three paths

(0, 0) -> (0, 1) -> (0, 2) -> (1, 2)

(0, 0) -> (0, 1) -> (1, 1) -> (1, 2)

(0, 0) -> (1, 0) -> (1, 1) -> (1, 2)

Count all possible paths: We can recursively move to right and down from the start until we reach the destination and then add up all valid paths to get the answer.

Procedure:

- Create a recursive function with parameters as row and column index
- Call this recursive function for N-1 and M-1
- In the recursive function
 - If N == 1 or M == 1 then return 1
 - else call the recursive function with (N-1, M) and (N, M-1) and return the sum of this
- Print the answer

Hints:

Python program to count all possible paths from top left to bottom right

Function to return count of possible paths to reach cell at row number m and column number n from the topmost leftmost cell (cell at 1, 1)

```
def numberOfPaths(m, n):
```

```
...
```

Driver program to test above function

```
m = 3
```

```
n = 3
```

```
print(numberOfPaths(m, n))
```

TRY

1. Take input : M = 3, N = 2 and verify the results.
2. Take input : M = 2, N = 1 and verify the results.

8. Graph Coloring

8.1 Graph Coloring using Greedy Algorithm

Greedy algorithm is used to assign colors to the vertices of a graph. It doesn't guarantee to use minimum colors, but it guarantees an upper bound on the number of colors. The basic algorithm never uses more than $d+1$ colors where d is the maximum degree of a vertex in the given graph.

Basic Greedy Coloring Algorithm:

1. Color first vertex with first color.
2. Do following for remaining $V-1$ vertices.
 - a) Consider the currently picked vertex and color it with the lowest numbered color that has not been used on any previously colored vertices adjacent to it. If all previously used colors appear on vertices adjacent to v , assign a new color to it.

Hints:

Implement greedy algorithm for graph coloring


```

def addEdge(adj, v, w):
    adj[v].append(w)
    # Note: the graph is undirected
    adj[w].append(v)
    return adj

# Assigns colors (starting from 0) to all
# vertices and prints the assignment of colors
def greedyColoring(adj, V):
    ...

# Driver Code
g1 = [[] for i in range(5)]
g1 = addEdge(g1, 0, 1)
g1 = addEdge(g1, 0, 2)
g1 = addEdge(g1, 1, 2)
g1 = addEdge(g1, 1, 3)
g1 = addEdge(g1, 2, 3)
g1 = addEdge(g1, 3, 4)
print("Coloring of graph 1 ")
greedyColoring(g1, 5)

g2 = [[] for i in range(5)]
g2 = addEdge(g2, 0, 1)
g2 = addEdge(g2, 0, 2)
g2 = addEdge(g2, 1, 2)
g2 = addEdge(g2, 1, 4)
g2 = addEdge(g2, 2, 4)
g2 = addEdge(g2, 4, 3)
print("\nColoring of graph 2")
greedyColoring(g2, 5)

```

Output:

Coloring of graph 1
Vertex 0 ---> Color 0
Vertex 1 ---> Color 1
Vertex 2 ---> Color 2
Vertex 3 ---> Color 0
Vertex 4 ---> Color 1

Coloring of graph 2
Vertex 0 ---> Color 0
Vertex 1 ---> Color 1
Vertex 2 ---> Color 2
Vertex 3 ---> Color 0
Vertex 4 ---> Color 3

8.2 Coloring a Cycle Graph

Given the number of vertices in a Cyclic Graph. The task is to determine the Number of colors required to color the graph so that no two adjacent vertices have the same color.

Approach:

- If the no. of vertices is Even then it is Even Cycle and to color such graph we require 2 colors.
- If the no. of vertices is Odd then it is Odd Cycle and to color such graph we require 3 colors.

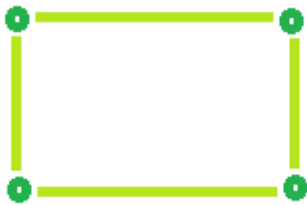
Input: Vertices = 3

Output: No. of colors require is: 3

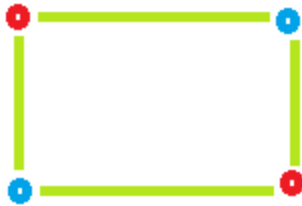
Input: vertices = 4

Output: No. of colors require is: 2

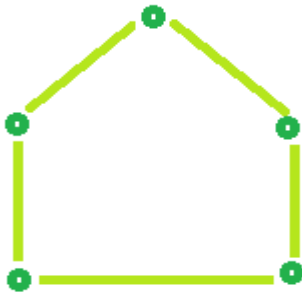
Example 1: Even Cycle: Number of vertices = 4



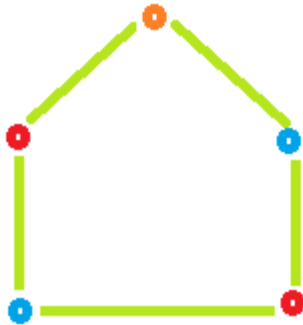
Color required = 2



Example 2: Odd Cycle: Number of vertices = 5



Color required = 3



Hints:

```
# Find the number of colors required to color a cycle graph
```

```
# Function to find Color required.
```

```
def Color(vertices):
```

```
...
```

```
# Driver Code
```

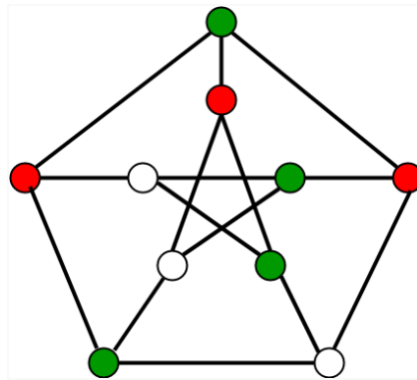
```
vertices = 3
```

```
print ("No. of colors require is:", Color(vertices))
```

8.3 m Coloring Problem

Given an undirected graph and a number m , determine if the graph can be colored with at most m colors such that no two adjacent vertices of the graph are colored with the same color.

Note: Here coloring of a graph means the assignment of colors to all vertices
Following is an example of a graph that can be colored with 3 different colors:



Input: graph = {0, 1, 1, 1},
 {1, 0, 1, 0},
 {1, 1, 0, 1},
 {1, 0, 1, 0}

Output: Solution Exists: Following are the assigned colors: 1 2 3 2

Explanation: By coloring the vertices with following colors, adjacent vertices does not have same colors

Input: graph = {1, 1, 1, 1},
 {1, 1, 1, 1},
 {1, 1, 1, 1},
 {1, 1, 1, 1}

Output: Solution does not exist

Explanation: No solution exists

Generate all possible configurations of colors. Since each node can be colored using any of the m available colors, the total number of color configurations possible is m^V . After generating a configuration of color, check if the adjacent vertices have the same color or not. If the conditions are met, print the combination and break the loop
Follow the given steps to solve the problem:

- Create a recursive function that takes the current index, number of vertices and output color array
- If the current index is equal to number of vertices. Check if the output color configuration is safe, i.e check if the adjacent vertices do not have same color. If the conditions are met, print the configuration and break
- Assign a color to a vertex (1 to m)
- For every assigned color recursively call the function with next index and number of vertices
- If any recursive function returns true break the loop and returns true.

Hints:

Number of vertices in the graph

define 4 4

check if the colored graph is safe or not

```
def isSafe(graph, color):
    # check for every edge
    for i in range(4):
        for j in range(i + 1, 4):
            if (graph[i][j] and color[j] == color[i]):
                return False
    return True
```

```
def graphColoring(graph, m, i, color):
    # write your code here
```

```

...

# /* A utility function to print solution */

def printSolution(color):
    print("Solution Exists:" " Following are the assigned colors ")
    for i in range(4):
        print(color[i], end=" ")

# Driver code
# /* Create following graph and test whether it is 3 colorable
# (3)---(2)
# | / |
# | / |
# | / |
# (0)---(1)
# */
graph = [
    [0, 1, 1, 1],
    [1, 0, 1, 0],
    [1, 1, 0, 1],
    [1, 0, 1, 0],
]
m = 3 # Number of colors

# Initialize all color values as 0.
# This initialization is needed
# correct functioning of isSafe()
color = [0 for i in range(4)]

# Function call
if (not graphColoring(graph, m, 0, color)):
    print("Solution does not exist")

```

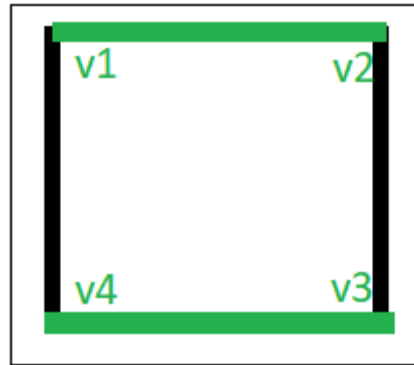
8.4 Edge Coloring of a Graph

Edge coloring of a graph is an assignment of “colors” to the edges of the graph so that no two adjacent edges have the same color with an optimal number of colors. Two edges are said to be adjacent if they are connected to the same vertex. There is no known polynomial time algorithm for edge-coloring every graph with an optimal number of colors.

Input: $u_1 = 1, v_1 = 4$
 $u_2 = 1, v_2 = 2$
 $u_3 = 2, v_3 = 3$
 $u_4 = 3, v_4 = 4$

Output: Edge 1 is of color 1
 Edge 2 is of color 2
 Edge 3 is of color 1
 Edge 4 is of color 2

The above input shows the pair of vertices (u_i, v_i) who have an edge between them. The output shows the color assigned to the respective edges.



Edge colorings are one of several different types of graph coloring problems. The above figure of a Graph shows an edge coloring of a graph by the colors green and black, in which no adjacent edge have the same color.

Algorithm:

1. Use BFS traversal to start traversing the graph.
2. Pick any vertex and give different colors to all of the edges connected to it, and mark those edges as colored.
3. Traverse one of it's edges.
4. Repeat step to with a new vertex until all edges are colored.

Hints:

Edge Coloring

```
from queue import Queue
```

function to determine the edge colors

```
def colorEdges(ptr, gra, edgeColors, isVisited):
    # Write your code here
    ...
```

Driver Function

```
empty=set()
# declaring vector of vector of pairs, to define Graph
gra=[]
edgeColors=[]
isVisited=[False]*100000

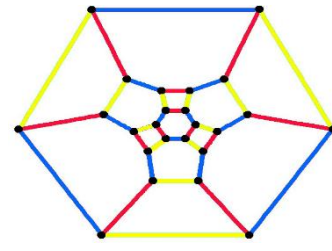
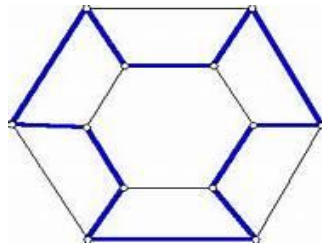
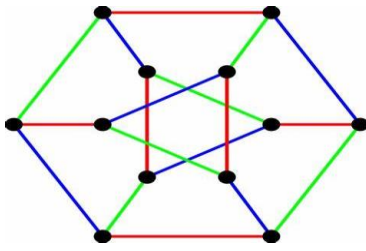
ver = 4
edge = 4
gra=[[ ] for _ in range(ver)]
edgeColors=[-1]*edge
gra[0].append((1, 0))
gra[1].append((0, 0))
gra[1].append((2, 1))
gra[2].append((1, 1))
gra[2].append((3, 2))
gra[3].append((2, 2))
gra[0].append((3, 3))
gra[3].append((0, 3))
colorEdges(0, gra, edgeColors, isVisited)
```

printing all the edge colors

```
for i in range(edge):
    print("Edge { } is of color { }".format(i + 1, edgeColors[i] + 1))
```

TRY

1. Write a program to implement graph coloring and edge coloring by consider the below graph and verify the results.



9. Graph Traversal

9.1 Breadth First Search

The **Breadth First Search (BFS)** algorithm is used to search a graph data structure for a node that meets a set of criteria. It starts at the root of the graph and visits all nodes at the current depth level before moving on to the nodes at the next depth level.

For a given graph G, print BFS traversal from a given source vertex.

Hints:

BFS traversal from a given source vertex.

```
from collections import defaultdict
```

This class represents a directed graph using adjacency list representation
class Graph:

Constructor

```
def __init__(self):
```

Default dictionary to store graph

```
self.graph = defaultdict(list)
```

Function to add an edge to graph

```
def addEdge(self, u, v):
```

```
self.graph[u].append(v)
```

Function to print a BFS of graph

```
def BFS(self, s):
```

```
    # Write your code here
```

```
    ...
```

Create a graph given in the above diagram

```
g = Graph()
```

```
g.addEdge(0, 1)
```

```
g.addEdge(0, 2)
```

```
g.addEdge(1, 2)
```

```
g.addEdge(2, 0)
```

```
g.addEdge(2, 3)
```

```
g.addEdge(3, 3)
```

```
print("Following is Breadth First Traversal" " (starting from vertex 2)")
```

```
g.BFS(2)
```

Output: Following is Breadth First Traversal (starting from vertex 2)

```
2 0 3 1
```

9.2 Depth First Search

Depth First Traversal (or DFS) for a graph is similar to Depth First Traversal of a tree. The only catch here is, that, unlike trees, graphs may contain cycles (a node may be visited twice). To avoid processing a node more than once, use a boolean visited array. A graph can have more than one DFS traversal.

For a given graph G, print DFS traversal from a given source vertex.

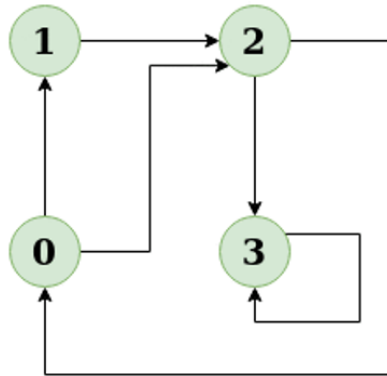
Input: n = 4, e = 6

0 -> 1, 0 -> 2, 1 -> 2, 2 -> 0, 2 -> 3, 3 -> 3

Output: DFS from vertex 1: 1 2 0 3

Explanation:

DFS Diagram:



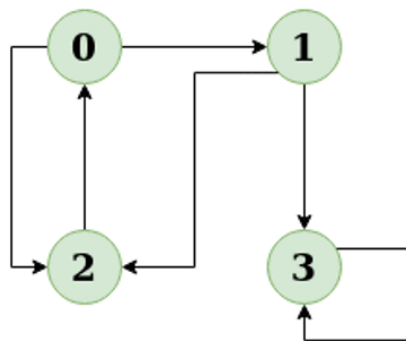
Input: n = 4, e = 6

2 -> 0, 0 -> 2, 1 -> 2, 0 -> 1, 3 -> 3, 1 -> 3

Output: DFS from vertex 2: 2 0 1 3

Explanation:

DFS Diagram:



Hints:

DFS traversal from a given graph
from collections import defaultdict

This class represents a directed graph using adjacency list representation
class Graph:

Constructor

```
def __init__(self):  
    # Default dictionary to store graph  
    self.graph = defaultdict(list)
```

Function to add an edge to graph

```
def addEdge(self, u, v):  
    self.graph[u].append(v)
```

A function used by DFS

```
def DFSUtil(self, v, visited):  
    ...
```

```
# The function to do DFS traversal. It uses recursive DFSUtil()
```

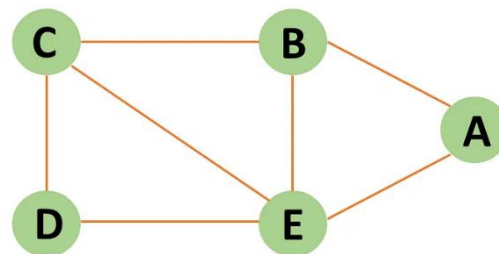
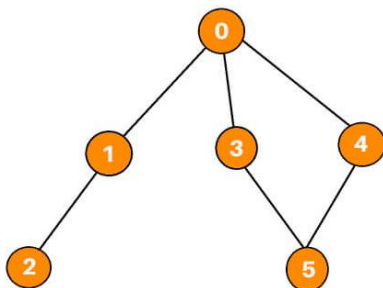
```
def DFS(self, v):  
    # Write your code here  
    ...
```

```
# Driver's code
```

```
g = Graph()  
g.addEdge(0, 1)  
g.addEdge(0, 2)  
g.addEdge(1, 2)  
g.addEdge(2, 0)  
g.addEdge(2, 3)  
g.addEdge(3, 3)  
print("Following is Depth First Traversal (starting from vertex 2)")  
# Function call  
g.DFS(2)
```

TRY

1. Write a program to implement breadth first search and depth first search by consider the below graph and verify the results.



10. Minimum Spanning Tree (MST)

10.1 Kruskal's Algorithm

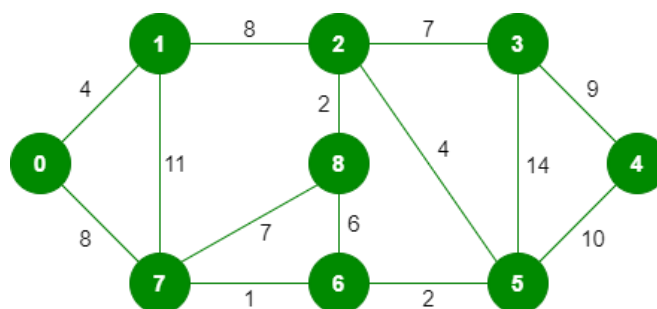
In Kruskal's algorithm, sort all edges of the given graph in increasing order. Then it keeps on adding new edges and nodes in the MST if the newly added edge does not form a cycle. It picks the minimum weighted edge at first and the maximum weighted edge at last.

MST using Kruskal's algorithm:

1. Sort all the edges in non-decreasing order of their weight.
2. Pick the smallest edge. Check if it forms a cycle with the spanning tree formed so far. If the cycle is not formed, include this edge. Else, discard it.
3. Repeat step#2 until there are $(V-1)$ edges in the spanning tree.

Kruskal's algorithm to find the minimum cost spanning tree uses the greedy approach. The Greedy Choice is to pick the smallest weight edge that does not cause a cycle in the MST constructed so far.

Input: For the given graph G find the minimum cost spanning tree.



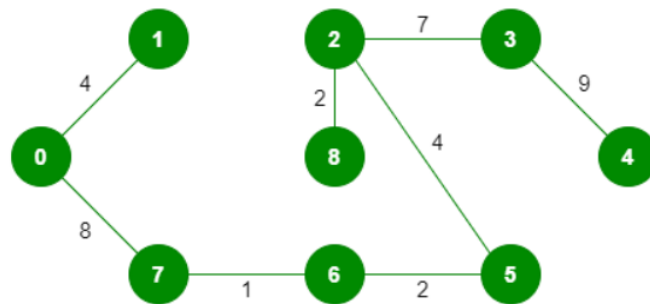
The graph contains 9 vertices and 14 edges. So, the minimum spanning tree formed will be having $(9 - 1) = 8$ edges.

After sorting:

Weight	Source	Destination
1	7	6
2	8	2
2	6	5
4	0	1
4	2	5
6	8	6
7	2	3
7	7	8
8	0	7
8	1	2
9	3	4
10	5	4
11	1	7
14	3	5

Now pick all edges one by one from the sorted list of edges.

Output:



Hints:

Kruskal's algorithm to find minimum Spanning Tree of a given connected,
undirected and weighted graph

Class to represent a graph

```
class Graph:
    def __init__(self, vertices):
        self.V = vertices
        self.graph = []
```

Function to add an edge to graph

```
def addEdge(self, u, v, w):
    self.graph.append([u, v, w])
```

```
def find(self, parent, i):
```

```
    ...
```

```
def union(self, parent, rank, x, y):
```

```
    ...
```

```
def KruskalMST(self):
```

```
    # write your code here
```

```
    ...
```

Driver code

```
g = Graph(4)
g.addEdge(0, 1, 10)
g.addEdge(0, 2, 6)
g.addEdge(0, 3, 5)
g.addEdge(1, 3, 15)
g.addEdge(2, 3, 4)
```

```
# Function call
g.KruskalMST()
```

Output: Following are the edges in the constructed MST

2 -- 3 == 4

0 -- 3 == 5

0 -- 1 == 10

Minimum Cost Spanning Tree: 19

10.2 Prim's Algorithm

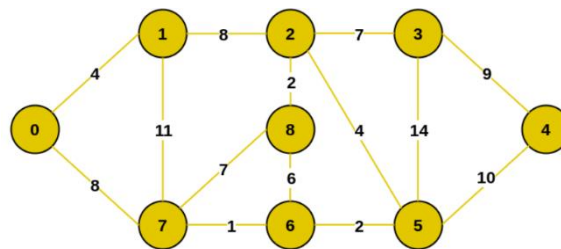
The Prim's algorithm starts with an empty spanning tree. The idea is to maintain two sets of vertices. The first set contains the vertices already included in the MST, and the other set contains the vertices not yet included. At every step, it considers all the edges that connect the two sets and picks the minimum weight edge from these edges. After picking the edge, it moves the other endpoint of the edge to the set containing MST.

Prim's Algorithm:

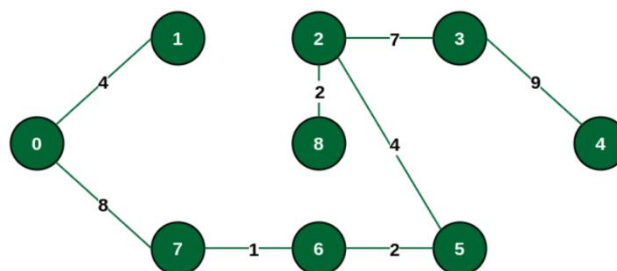
The working of Prim's algorithm can be described by using the following steps:

1. Determine an arbitrary vertex as the starting vertex of the MST.
2. Follow steps 3 to 5 till there are vertices that are not included in the MST (known as fringe vertex).
3. Find edges connecting any tree vertex with the fringe vertices.
4. Find the minimum among these edges.
5. Add the chosen edge to the MST if it does not form any cycle.
6. Return the MST and exit

Input: For the given graph G find the minimum cost spanning tree.



Output: The final structure of the MST is as follows and the weight of the edges of the MST is $(4 + 8 + 1 + 2 + 4 + 2 + 7 + 9) = 37$.



Hints:

```
# Prim's Minimum Spanning Tree (MST) algorithm.
# The program is for adjacency matrix representation of the graph
```

```
# Library for INT_MAX
import sys
```

```
class Graph():
```

```

def __init__(self, vertices):
    self.V = vertices
    self.graph = [[0 for column in range(vertices)]
                  for row in range(vertices)]

# A utility function to print
# the constructed MST stored in parent[]
def printMST(self, parent):
    print("Edge \tWeight")
    for i in range(1, self.V):
        print(parent[i], "-", i, "\t", self.graph[i][parent[i]])

# A utility function to find the vertex with
# minimum distance value, from the set of vertices
# not yet included in shortest path tree
def minKey(self, key, mstSet):

    ...

def primMST(self):
    ...

# Driver's code
g = Graph(5)
g.graph = [[0, 2, 0, 6, 0],
           [2, 0, 3, 8, 5],
           [0, 3, 0, 0, 7],
           [6, 8, 0, 0, 9],
           [0, 5, 7, 9, 0]]

g.primMST()

```

Output:

Edge Weight

0 - 1 2

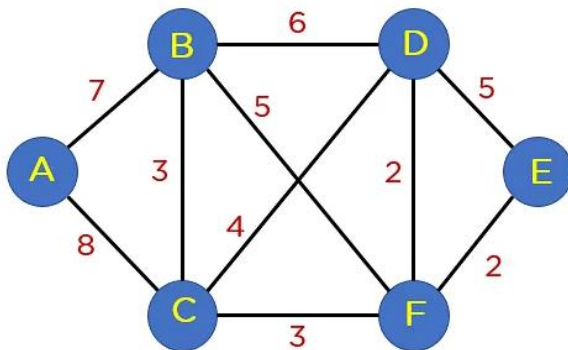
1 - 2 3

0 - 3 6

1 - 4 5

TRY

1. Write a program to implement kruskal's algorithm and prim's algorithm by consider the below graph and verify the results.



11. Roots of Equations

11.1 Bisection Method

The Bisection method is also called the interval halving method, the binary search method or the dichotomy method. This method is used to find root of an equation in a given interval that is value of 'x' for which $f(x) = 0$. The method is based on **The Intermediate Value Theorem** which states that if $f(x)$ is a continuous function and there are two real

numbers a and b such that $f(a) * f(b) < 0$ and $f(b) < 0$, then it is guaranteed that it has at least one root between them.

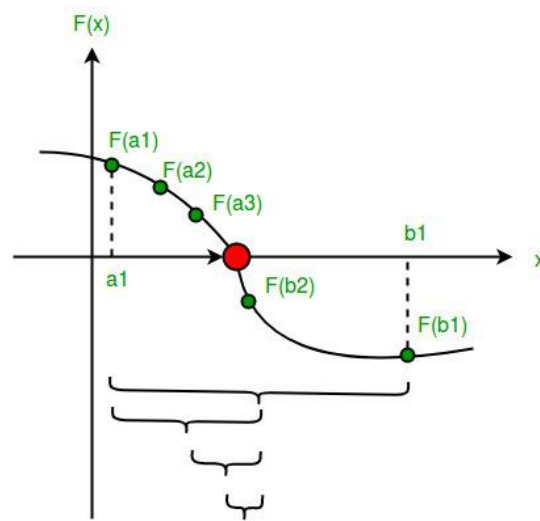
Assumptions:

1. $f(x)$ is a continuous function in interval $[a, b]$
2. $f(a) * f(b) < 0$

Steps:

1. Find middle point $c = (a + b)/2$.
2. If $f(c) == 0$, then c is the root of the solution.
3. Else $f(c) != 0$
 - If value $f(a)*f(c) < 0$ then root lies between a and c . So we recur for a and c
 - Else If $f(b)*f(c) < 0$ then root lies between b and c . So we recur b and c .
 - Else given function doesn't follow one of assumptions.

Since root may be a floating point number, we repeat above steps while difference between a and b is greater than and equal to a value? (A very small value).



Hints:

An example function whose solution is determined using Bisection Method.

The function is $x^3 - x^2 + 2$

```
def func(x):
    return x*x*x - x*x + 2
```

Prints root of func(x) with error of EPSILON

```
def bisection(a,b):
    # Write your code here
    ...
```

Driver code

Initial values assumed

$a = -200$

$b = 300$

bisection (a, b)

Output: The value of root is : -1.0025

TRY

1. Take an input function $x^2 - x^3 + 2$ and verify the results.
2. Take an input function $x^3 - x^3 + 4$ and verify the results.

11.2 Method of False Position

Given a function $f(x)$ on floating number x and two numbers ' a ' and ' b ' such that $f(a)*f(b) < 0$ and $f(x)$ is continuous in $[a, b]$. Here $f(x)$ represents algebraic or transcendental equation. Find root of function in interval $[a, b]$ (Or find a

value of x such that $f(x)$ is 0).

Input: A function of x , for example $x^3 - x^2 + 2$.
And two values: $a = -200$ and $b = 300$ such that
 $f(a) \cdot f(b) < 0$, i.e., $f(a)$ and $f(b)$ have opposite signs.

Output: The value of root is : -1.00
OR any other value close to root.

Hints:

```
MAX_ITER = 1000000

# An example function whose solution is determined using Regular Falsi Method.
# The function is x^3 - x^2 + 2
def func( x ):
    return (x * x * x - x * x + 2)

# Prints root of func(x) in interval [a, b]
def regulaFalsi( a , b):
    # Write your code here
    ...

# Driver code to test above function
# Initial values assumed
a = -200
b = 300
regulaFalsi(a, b)
```

TRY

1. Take an input function $x^2 - x^3 + 2$ and verify the results.
2. Take an input function $x^3 - x^3 + 4$ and verify the results.

11.3 Newton Raphson Method

Given a function $f(x)$ on floating number x and an initial guess for root, find root of function in interval. Here $f(x)$ represents algebraic or transcendental equation.

Input: A function of x (for example $x^3 - x^2 + 2$), derivative function of x ($3x^2 - 2x$ for above example) and an initial guess $x_0 = -20$

Output: The value of root is: -1.00 or any other value close to root.

Algorithm:

Input: initial x , $\text{func}(x)$, $\text{derivFunc}(x)$

Output: Root of $\text{Func}()$

1. Compute values of $\text{func}(x)$ and $\text{derivFunc}(x)$ for given initial x
2. Compute h : $h = \text{func}(x) / \text{derivFunc}(x)$
3. While h is greater than allowed error ϵ
 - $h = \text{func}(x) / \text{derivFunc}(x)$
 - $x = x - h$

Hints:

```
# Implementation of Newton Raphson Method for solving equations
# An example function whose solution is determined using Bisection Method.
# The function is x^3 - x^2 + 2
def func( x ):
    return x * x * x - x * x + 2

# Derivative of the above function which is 3*x^2 - 2*x
def derivFunc( x ):
    return 3 * x * x - 2 * x
```

```
# Function to find the root
def newtonRaphson( x ):
    # Write your code here
    ...

# Driver program
x0 = -20
newtonRaphson(x0)
```

TRY

1. Take an input function $x^2 - x^3 + 2$ and verify the results.
2. Take an input function $x^3 - x^3 + 4$ and verify the results.

11.4 Secant Method

The secant method is used to find the root of an equation $f(x) = 0$. It is started from two distinct estimates x_1 and x_2 for the root. It is an iterative procedure involving linear interpolation to a root. The iteration stops if the difference between two intermediate values is less than the convergence factor.

Input: Equation = $x^3 + x - 1$
 $x_1 = 0, x_2 = 1, E = 0.0001$

Output: Root of the given equation = 0.682326
 No. of iteration=5

Algorithm

Initialize: x_1, x_2, E, n // E = convergence indicator
 calculate $f(x_1), f(x_2)$

```
if(f(x1) * f(x2) = E); //repeat the loop until the convergence
    print 'x0' //value of the root
    print 'n' //number of iteration
}
else
    print "cannot found a root in the given interval"
```

Hints:

```
# Find root of an equations using secant method
# Function takes value of x and returns f(x)
def f(x):
    # we are taking equation as x^3+x-1
    f = pow(x, 3) + x - 1;
    return f;

def secant(x1, x2, E):
    # Write your code here
    ...

# Driver code
# initializing the values
x1 = 0;
x2 = 1;
E = 0.0001;
secant(x1, x2, E);
```

TRY

1. Take an input function $x^2 - x + 2$ and verify the results.
2. Take an input function $x^3 - x^2 + 4$ and verify the results.

11.5 Muller Method

Given a function $f(x)$ on floating number x and three initial distinct guesses for root of the function, find the root of function. Here, $f(x)$ can be an algebraic or transcendental function.

Input: A function $f(x) = x^3 + 2x^2 + 10x - 20$ and three initial guesses - 0, 1 and 2 .

Output: The value of the root is 1.3688 or any other value within permissible deviation from the root.

Input: A function $f(x) = x^3 - 5x + 2$ and three initial guesses - 0, 1 and 2.

Output: The value of the root is 0.4021 or any other value within permissible deviation from the root.

Hints:

```
# Find root of a function, f(x)
import math;

MAX_ITERATIONS = 10000;

# Function to calculate f(x)
def f(x):
    # Taking f(x) = x^3 + 2x^2 + 10x - 20
    return (1 * pow(x, 3) + 2 * x * x +
            10 * x - 20);

def Muller(a, b, c):
    # Write your code here
    ...

# Driver Code
a = 0;
b = 1;
c = 2;
Muller(a, b, c);
```

TRY

1. Take an input function $x^2 - x^3 + 2$ and verify the results.
2. Take an input function $x^3 - x^3 + 4$ and verify the results.

12. Numerical Integration

12.1 Trapezoidal Rule for Approximate Value of Definite Integral

Trapezoidal rule is used to find the approximation of a definite integral. The basic idea in Trapezoidal rule is to assume the region under the graph of the given function to be a trapezoid and calculate its area.

$$\int_a^b f(x) dx \approx (b - a) \left[\frac{f(a) + f(b)}{2} \right]$$

Hints:

```
# Implement Trapezoidal rule

# A sample function whose definite integral's approximate value is
# computed using Trapezoidal rule
def y( x ):

    # Declaring the function
    # f(x) = 1/(1+x*x)
    return (1 / (1 + x * x))

# Function to evaluate the value of integral
def trapezoidal (a, b, n):
    # Write your code here
```

```

...

# Driver code
# Range of definite integral
x0 = 0
xn = 1

# Number of grids. Higher value means more accuracy
n = 6
print ("Value of integral is ",
      "%.4f"%trapezoidal(x0, xn, n))

```

12.2 Simpson's 1/3 Rule

Simpson's 1/3 rule is a method for numerical approximation of definite integrals. Specifically, it is the following approximation:

$$\int_a^b f(x)dx \approx \frac{(b-a)}{6} \left(f(a) + 4f\frac{(a+b)}{2} + f(b) \right)$$

Procedure:

In order to integrate any function f(x) in the interval (a, b), follow the steps given below:

1. Select a value for n, which is the number of parts the interval is divided into.
2. Calculate the width, $h = (b-a)/n$
3. Calculate the values of x0 to xn as $x_0 = a$, $x_1 = x_0 + h$, ..., $x_{n-1} = x_{n-2} + h$, $x_n = b$.
Consider $y = f(x)$. Now find the values of y (y_0 to y_n) for the corresponding x (x_0 to x_n) values.
4. Substitute all the above found values in the Simpson's Rule Formula to calculate the integral value.

Approximate value of the integral can be given by **Simpson's Rule:**

$$\int_a^b f(x)dx \approx \frac{h}{3} \left(f_0 + f_n + 4 * \sum_{i=1,3,5}^{n-1} f_i + 2 * \sum_{i=2,4,6}^{n-2} f_i \right)$$

Input: Evaluate $\log x$ dx within limit 4 to 5.2.

First we will divide interval into six equal parts as number of interval should be even.

x : 4 4.2 4.4 4.6 4.8 5.0 5.2

\log_x : 1.38 1.43 1.48 1.52 1.56 1.60 1.64

Output: Now we can calculate approximate value of integral using above formula:

$$\begin{aligned}
 &= h/3[(1.38 + 1.64) + 4 * (1.43 + 1.52 + 1.60) + 2 *(1.48 + 1.56)] \\
 &= 1.84
 \end{aligned}$$

Hence the approximation of above integral is

1.827 using Simpson's 1/3 rule.

Hints:

```

# Simpson's 1 / 3 rule
import math

# Function to calculate f(x)
def func(x):
    return math.log(x)

# Function for approximate integral

```



```
def simpsons_(ll, ul, n):
    # Write your code here
    ...

# Driver code
lower_limit = 4    # Lower limit
upper_limit = 5.2  # Upper limit
n = 6              # Number of interval
print("%.6f"% simpsons_(lower_limit, upper_limit, n))
```

12.3 Simpson's 3/8 Rule

The Simpson's 3/8 rule was developed by Thomas Simpson. This method is used for performing numerical integrations. This method is generally used for numerical approximation of definite integrals. Here, parabolas are used to approximate each part of curve.

Input: lower_limit = 1, upper_limit = 10, interval_limit = 10

Output: integration_result = 0.687927

Input: lower_limit = 1, upper_limit = 5, interval_limit = 3

Output: integration_result = 0.605835

Hints:

```
# Implement Simpson's 3/8 rule

# Given function to be integrated
def func(x):
    return (float(1) / ( 1 + x * x ))

# Function to perform calculations
def calculate(lower_limit, upper_limit, interval_limit):
    # Write your code here
    ...

# driver function
interval_limit = 10
lower_limit = 1
upper_limit = 10

integral_res = calculate(lower_limit, upper_limit, interval_limit)

# rounding the final answer to 6 decimal places
print (round(integral_res, 6))
```

13. Ordinary Differential Equations

13.1 The Euler Method

Given a differential equation $dy/dx = f(x, y)$ with initial condition $y(x_0) = y_0$. Find its approximate solution using Euler method.

Euler Method:

In mathematics and computational science, the Euler method (also called forward Euler method) is a first-order numerical procedure for solving ordinary differential equations (ODEs) with a given initial value.

Consider a differential equation $dy/dx = f(x, y)$ with initial condition $y(x_0) = y_0$ then a successive approximation of this equation can be given by:

$$y(n+1) = y(n) + h * f(x(n), y(n))$$

where $h = (x(n) - x(0)) / n$, h indicates step size. Choosing smaller values of h leads to more accurate results and more computation time.

Example:

Consider below differential equation $dy/dx = (x + y + xy)$ with initial condition $y(0) = 1$ and step size $h = 0.025$. Find $y(0.1)$.

Solution:

$$f(x, y) = (x + y + xy)$$

$$x_0 = 0, y_0 = 1, h = 0.025$$

Now we can calculate y_1 using Euler formula

$$y_1 = y_0 + h * f(x_0, y_0)$$

$$y_1 = 1 + 0.025 * (0 + 1 + 0 * 1)$$

$$y_1 = 1.025$$

$$y(0.025) = 1.025.$$

Similarly we can calculate $y(0.050)$, $y(0.075)$, ..., $y(0.1)$.

$$y(0.1) = 1.11167$$

Hints:

Find approximation of an ordinary differential equation using Euler method.

Consider a differential equation

$dy / dx = (x + y + xy)$

def func(x, y):

 return (x + y + x * y)

Function for Euler formula

def euler(x0, y, h, x):

 # write code here

 ...

Driver Code

Initial Values

$$x_0 = 0$$

$$y_0 = 1$$

$$h = 0.025$$

Value of x at which we need approximation

$$x = 0.1$$

$$\text{euler}(x_0, y_0, h, x)$$

13.2 Runge-Kutta Second Order Method

Given the following inputs:

1. An ordinary differential equation that defines the value of dy/dx in the form x and y .

$$\frac{dy}{dx} = f(x, y)$$

2. Initial value of y , i.e., $y(0)$.

$$y(0) = y_0$$

The task is to find the value of unknown function y at a given point x , i.e. $y(x)$.

Input: $x_0 = 0, y_0 = 1, x = 2, h = 0.2$

Output: $y(x) = 0.645590$

Input: $x_0 = 2, y_0 = 1, x = 4, h = 0.4$;

Output: $y(x) = 4.122991$

Approach:

The Runge-Kutta method finds an approximate value of y for a given x . Only first-order ordinary differential equations can be solved by using the Runge-Kutta 2nd-order method.

Below is the formula used to compute the next value y_{n+1} from the previous value y_n . Therefore:

$$y_{n+1} = \text{value of } y \text{ at } (x = n + 1)$$

$$y_n = \text{value of } y \text{ at } (x = n)$$

where $0 \leq n \leq (x - x_0)/h$, h is step height

$$x_{n+1} = x_0 + h$$

The essential formula to compute the value of y_{n+1} :

$$K1 = h * f(x, y)$$

$$K2 = h * f(x/2, y/2) \text{ or } K1/2$$

$$y_{n+1} = y_n + K^2 + (h^3)$$

The formula basically computes the next value y_{n+1} using current y_n plus the weighted average of two increments:

- **K1** is the increment based on the slope at the beginning of the interval, using y .
- **K2** is the increment based on the slope at the midpoint of the interval, using $(y + h*K1/2)$.

Hints:

```
# Implement Runge-Kutta method
```

```
# A sample differential equation
```

```
# "dy/dx = (x - y)/2"
```

```
def dydx(x, y):
    return (x + y - 2)
```

```
# Finds value of y for a given x using step size h and initial value y0 at x0.
```

```
def rungeKutta(x0, y0, x, h):
```

```
    # write code here
```

```
    ...
```

```
# Driver Code
```

```
x0 = 0
```

```
y = 1
```

```
x = 2
```

```
h = 0.2
```

```
print("y(x) =", rungeKutta(x0, y, x, h))
```

14. Final Notes

The only way to learn programming is program, program and program on challenging problems. The problems in this tutorial are certainly NOT challenging. There are tens of thousands of challenging problems available – used in training for various programming contests (such as International Collegiate Programming Contest (ICPC), International Olympiad in Informatics (IOI)). Check out these sites:

- The ACM - ICPC International collegiate programming contest (<https://icpc.global/>)
- The Topcoder Open (TCO) annual programming and design contest (<https://www.topcoder.com/>)
- Universidad de Valladolid's online judge (<https://uva.onlinejudge.org/>).
- Peking University's online judge (<http://poj.org/>).
- USA Computing Olympiad (USACO) Training Program @ <http://train.usaco.org/usacogate>.
- Google's coding competitions (<https://codingcompetitions.withgoogle.com/codejam>, <https://codingcompetitions.withgoogle.com/hashcode>)
- The ICFP programming contest (<https://www.icfpconference.org/>)
- BME International 24-hours programming contest (<https://www.challenge24.org/>)
- The International Obfuscated C Code Contest (<https://www0.us.ioccc.org/main.html>)
- Internet Problem Solving Contest (<https://ipsc.ksp.sk/>)
- Microsoft Imagine Cup (<https://imaginecup.microsoft.com/en-us>)
- Hewlett Packard Enterprise (HPE) Codewars (<https://hpecodewars.org/>)
- OpenChallenge (<https://www.openchallenge.org/>)

Coding Contests Scores

Students must solve problems and attain scores in the following coding contests:

	Name of the contest	Minimum number of problems to solve	Required score
•	CodeChef	20	200
•	Leetcode	20	200
•	GeeksforGeeks	20	200

• SPOJ	5	50
• InterviewBit	10	1000
• Hackerrank	25	250
• Codeforces	10	100
• BuildIT	50	500
Total score need to obtain		2500

Student must have any one of the following certification:

1. HackerRank - Problem Solving Skills Certification (Basic and Intermediate)
2. GeeksforGeeks – Data Structures and Algorithms Certification
3. CodeChef - Learn Python Certification
4. Interviewbit – DSA pro / Python pro
5. NPTEL – Programming, Data Structures and Algorithms
6. NPTEL – The Joy of Computing using Python

V. TEXT BOOKS:

1. Eric Matthes, “Python Crash Course: A Hands-On, Project-based Introduction to Programming”, No Starch Press, 3rd Edition, 2023.
2. John M Zelle, “Python Programming: An Introduction to Computer Science”, Ingram short title, 3rd Edition, 2016.

VI. REFERENCE BOOKS:

1. Yashavant Kanetkar, Aditya Kanetkar, “Let Us Python”, BPB Publications, 2nd Edition, 2019.
2. Martin C. Brown, “Python: The Complete Reference”, Mc. Graw Hill, Indian Edition, 2018.
3. Paul Barry, “Head First Python: A Brain-Friendly Guide”, O’Reilly, 2nd Edition, 2016
4. Taneja Sheetal, Kumar Naveen, “Python Programming – A Modular Approach”, Pearson, 1st Edition, 2017.
5. R Nageswar Rao, “Core Python Programming”, Dreamtech Press, 2018.

VII. ELECTRONIC RESOURCES

8. <https://realPython.com/Python3-object-oriented-programming/>
9. <https://Python.swaroopch.com/oop.html>
10. https://Python-textbok.readthedocs.io/en/1.0/Object_Oriented_Programming.html
11. <https://www.programiz.com/Python-programming/>
12. <https://www.geeksforgeeks.org/python-programming-language/>

VIII. MATERIALS ONLINE

1. Course template
2. Lab Manual



INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal - 500 043, Hyderabad, Telangana

COURSE CONTENT

ENGINEERING GRAPHICS								
I Semester: CSE / CSE(DS) / CSE(CS)								
II Semester: ECE / EEE / CSE (AI&ML) / IT								
Course Code	Category	Hours / Week			Credits	Maximum Marks		
AMED03	Foundation	L	T	P	C	CIA	SEE	Total
		0	1	2	2	40	60	100
Contact Classes: 15	Tutorial Classes: Nil	Practical Classes: 30			Total Classes:45			
Prerequisite: Fundamentals of Geometry								

I. COURSE OVERVIEW:

Introduction to graphical representation using free hand drawing and computer-aided drafting. Engineering graphics covers basic engineering drawing techniques such as lines & lettering, geometrical constructions, principles of tangency, orthographic projections, sectional views, and dimensioning. This course assists to draw 2D drawings for industrial applications.

II. COURSES OBJECTIVES:

The students will try to learn

1. The basic engineering drawing formats.
2. Projections of points, lines, planes and solids at inclinations of horizontal plane and vertical plane.
3. Use of computer-aided design (CAD) to communicate concepts and ideas in the design of three-dimensional engineering products.

III. COURSE OUTCOMES:

At the end of the course students should be able to:

- CO1 Demonstrate an ability to dimension and annotate two-dimensional engineering graphics.
- CO2 Demonstrate the freehand sketching to aid in the visualization process and to efficiently communicate ideas graphically.
- CO3 Make use of CAD software for the creation of 3D models and 2D engineering graphics.
- CO4 Comprehend the principles and techniques for creating sectional views of three-dimensional solids in engineering graphics.
- CO5 Explain the application of industry standards and best practices applied in engineering graphics.
- CO6 Apply the general projection theory with emphasis on orthographic projection to represent three-dimensional objects in two-dimensional views

IV. COURE CONTENT:

EXCERCISES ON ENGINEERING GRAPHICS

Note: Students are encouraged to bring their own laptops for laboratory practice sessions.

1. Getting Started Exercises

1.1 Introduction to AUTOCAD

AutoCAD is a widely-used computer-aided design (CAD) software application developed by Autodesk. It has been an industry standard for drafting and designing since its inception in the early 1980s. AutoCAD provides a versatile platform for creating and editing 2D and 3D drawings and models, making it an essential tool in various fields such as architecture, engineering, construction, manufacturing, and more.

- i. Install AUTO CAD
- ii. Purpose and Application
- iii. Interface and Tools
- iv. Precision and Accuracy
- v. 2D and 3D Modeling
- vi. Collaboration and Sharing
- vii. Customization
- viii. Industry Usage
- ix. Versions and Licensing

1.1 Commands

The main purpose of using commands and shortcuts in AutoCAD boils down to increased productivity. They allow you to execute functions more quickly, as you don't need to search through the entire AutoCAD interface for the right tool. You can just type the command, and the function window appears.

- i. Basic Drawing Commands
- ii. Editing Commands
- iii. Dimensioning Commands
- iv. Advanced and Miscellaneous Commands

Basic Drawing Commands:

- Line (LINE): Draws straight line segments between two points.
- Circle (CIRCLE): Creates circles by specifying a center point and radius.
- Rectangle (RECTANGLE): Constructs rectangles by defining two opposing corners.
- Arc (ARC): Draws arcs based on different methods, such as specifying start, end, and radius or center, start, and angle.

Editing Commands:

- Erase (ERASE): Deletes selected objects from the drawing.
- Copy (COPY): Copies objects to a specified location.
- Move (MOVE): Relocates selected objects to a different position.
- Trim (TRIM): Cuts selected objects at the cutting edges defined by other objects.
- Extend (EXTEND): Extends objects to meet the boundaries of other objects.

Dimensioning Commands:

- Line Dimension (DIMLINEAR): Adds linear dimensions to objects.
- Aligned Dimension (DIMALIGNED): Creates dimensions aligned with an angle of the object.
- Radial Dimension (DIMRADIUS): Add radius dimensions to arcs and circles.
- Diameter Dimension (DIMDIAMETER): Creates diameter dimensions for circles.

Advanced and Miscellaneous Commands:

- Hatch (HATCH): Fills enclosed areas with a pattern or gradient.
- Offset (OFFSET): Creates parallel copies of objects at a specified distance.
- Block (BLOCK): Defines reusable blocks (collections of objects) in the drawing.
- Insert (INSERT): Inserts predefined blocks into the drawing.
- Viewport (VPORST): Manages viewports for layout and plotting in paper space.
- Layer (LAYER): Manages layers for organizing and controlling object visibility.

TRY: Observe Exercise 1.1 in Solid works and in Creo software.

2. Introduction to Engineering Drawing

Engineering drawing, often referred to as technical drawing or drafting, is a graphical representation of an object, system, or structure used in various fields of engineering, manufacturing, and architecture. These drawings serve as a universal language that communicates design ideas, specifications, and instructions in a precise and standardized manner.

2.1 Basic Exercises

To be proficient in engineering drawing, basic exercises are required.

- i. Identify the basic tools used for drafting
- ii. Types of lines
- iii. Arcs
- iv. Circles

2.2 Practicing the standard lettering and numbering

Practicing standard lettering and numbering in engineering drawing is crucial for creating clear, professional, and easily understandable technical drawings. Proper lettering and numbering enhance communication and ensure that your drawings convey information accurately.

The following exercises are to be practiced to become proficient in lettering and numbering.

1. Use the correct fonts
2. Maintain uniformity
3. Lettering style
4. Height and spacing

Try: The following questions are to be answered in Solid works

1. How to use correct fonts in Solid Works
2. What are the commands are used to maintain uniformity of lettering and numbering.

3. Dimensioning

Dimensioning in engineering drawing is a crucial aspect that involves adding measurements and annotations to convey the size, location, and tolerances of objects, features, and components accurately. Proper dimensioning is essential for manufacturing, construction, and other engineering processes.

3.1 Exercises on Dimensioning

1. Understanding and use of the conventional dimensioning techniques.
2. Placing the dimension lines
3. Extension lines
4. Dimensions on angles

Hint: 1. The following Fig.3.1 shows the type of dimensioning on 2D drawing.

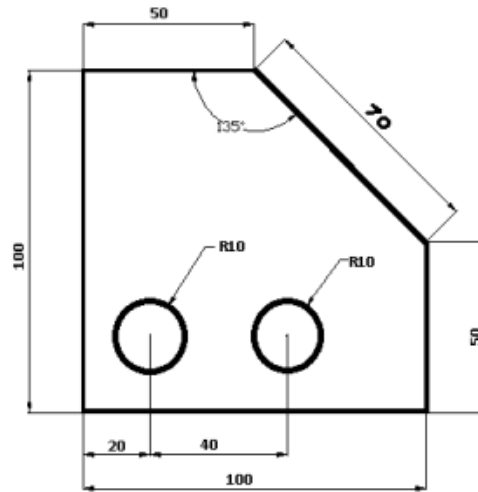


Fig.3.1 Dimensioning on 2D drawing

Hint: 2. The following Fig.3.2 shows the type of dimensioning on concentric circles.

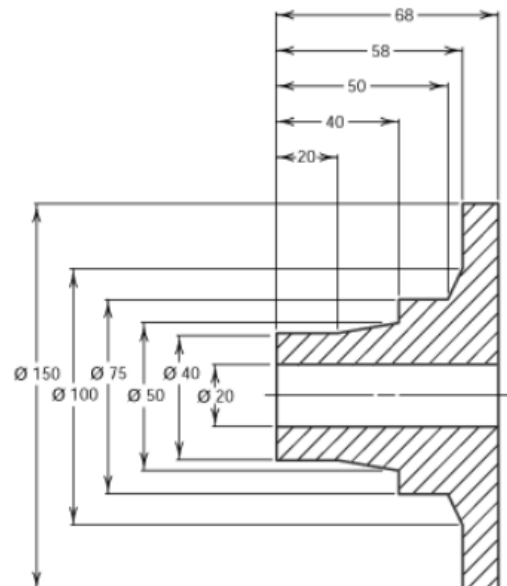


Fig.3.2 Dimensioning on Concentric circle

Try: Demonstrate the exercise 3 in Solid Works and CREO software.

4. Geometrical Constructions

Geometric construction is useful for learning how to use geometric tools like a ruler, compass, and straightedge to draw various angles, line segments, bisectors, and other forms of polygons, arcs, circles, and other geometric figures. Fig.4.1 shows the various geometric shapes to draw orthographic projections of lines, planes and solids.

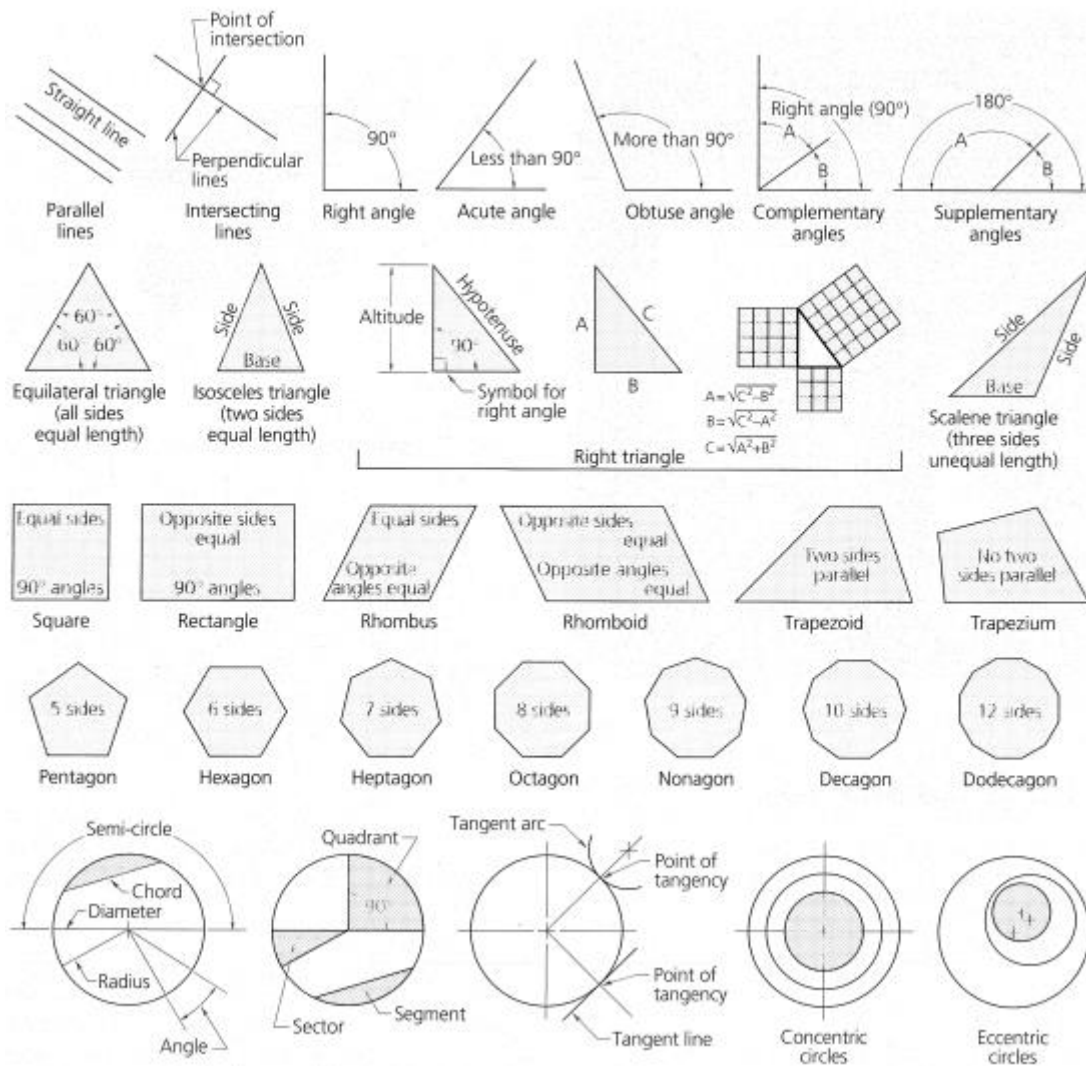


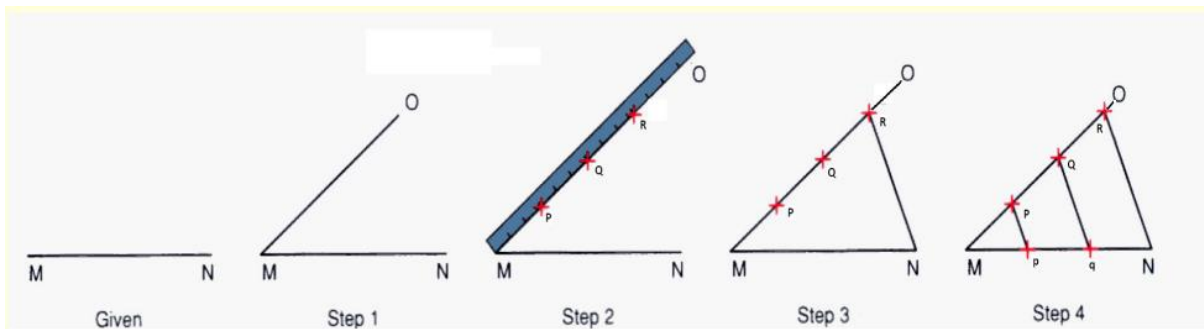
Fig.4.1 Geometric shapes

4.1. Exercises on Geometrical Constructions

To become proficient in engineering graphics geometrical constructions are required: Drawing lines, angles, triangle, square, pentagon, hexagon, octagon. Dividing line into equal or proportional parts. Drawing lines and arcs tangent to each other.

1. Divide a 16 cm straight line into a given number of equal parts say 5.
2. Divide a 8 cm line into 9 number of parts.
3. Bisect a given 45 degree sector.
4. Bisect a given straight line.
5. To draw a perpendicular to a given line from a point within it.
6. Construct a regular polygon, given the length of its side.

Hint: Dividing a line into equal number of parts



Try: The exercise 4 in Solid Works and CREO software.

5. Conic Sections

A conic section, conic or a quadratic curve is a curve obtained from a cone's surface intersecting a plane. The three types of conic section are the hyperbola, the parabola, and the ellipse.

5.1. Exercises on Conic Sections

1. Draw an ellipse with the distance of the focus from the directrix at 50mm and eccentricity = $2/3$ (Eccentricity method)
2. Draw an ellipse with the distance of the focus from the directrix at 60mm and eccentricity = $2/3$ (Eccentricity method)
3. Draw an ellipse with the distance of the focus from the directrix at 80mm and eccentricity = $2/3$ (Eccentricity method)
4. Draw a parabola with the distance of the focus from the directrix at 50 mm (Eccentricity method).
5. Draw a parabola with the distance of the focus from the directrix at 40mm (Eccentricity method).
6. A vertex of a hyperbola is 60 mm from its focus. Draw two parts of the hyperbola; if the eccentricity is $3/2$.
7. A vertex of a hyperbola is 50 mm from its focus. Draw two parts of the hyperbola; if the eccentricity is 1.5.

Try: The exercise 5 in Solid Works and CREO software.

6. Technical Sketching and Shape Description

6.1. Projections of planes and regular solids

1. Draw the projections of a regular pentagon of 30 mm side, having its surface inclined at 30° to the H.P. and a side parallel to the H.P. and inclined at an angle of 60° to the V.P.
2. Draw the projections of a regular hexagon of 40 mm side, having one of its sides in the H.P. and inclined at 60° to the V.P., and its surface making an angle of 45° with the H.P.
3. Draw the projections of a regular hexagon of 35 mm side, having one of its sides in the H.P. and inclined at 50° to the V.P., and its surface making an angle of 45° with the H.P.
4. Draw the projections of a regular pentagon of 40 mm side, having one of its sides in the H.P. and inclined at 30° to the V.P., and its surface making an angle of 45° with the H.P.

Try: The exercise 6.3 in Solid Works and CREO software.

7. Sectional views

A sectional view represents the part of an object remaining after a portion is assumed to have been cut and removed. The exposed cut surface is then indicated by section lines. Hidden features behind the cutting plane are omitted, unless required for dimensioning or for definition of the part.

7.1. Exercise on Sectional views of right regular solids, prism, cylinder, pyramid, cone.

1. A pentagonal pyramid, base 40 mm side and axis 60 mm long has its base horizontal and an edge of the base parallel to the V.P. A horizontal section plane cuts it at a distance of 20 mm above the base. Draw its front view and sectional top view.
2. A hexagonal prism, side of base 40 mm and height 70 mm is resting on one of its corners on the H.P. with a longer edge containing that corner inclined at 40° to the H.P. and a rectangular face parallel to the V.P. Draw the front view and sectional top view of the cut prism when a horizontal

section plane cuts the prism in two equal halves. Draw the front view and sectional top view of the cut prism.

3. A pentagonal pyramid, base 40 mm side and axis 70 mm long has one of its triangular faces in the V.P. and the edge of the base contained by that face makes an angle of 40° with the H.P. Draw its projections.
4. Draw the projections of a cone, base 50 mm diameter and axis 75 mm long, lying on a generator on the ground with the top view of the axis making an angle of 45° with the V.P.

Try: The exercise 7.1 and 7.2 in Solid Works and CREO software.

8. Development of surfaces

Knowledge of development is very useful in sheet metal work, construction of storage vessels, chemical vessels, boilers, and chimneys. Such vessels are manufactured from plates that are cut according to these developments and then properly bend into desired shaped.

8.1. Exercise on Basics of development of surfaces

1. Draw the development of the lateral surfaces of a right square prism of edge of base 30 mm and axis 50 mm long.
2. Draw the development of the complete surface of a cylindrical drum. Diameter is 40 mm and height 60 mm.

8.2. Exercise on Development of surfaces of Prisms

1. A hexagonal prism of base side 20 mm and height 45 mm is resting on one of its ends on the HP with two of its lateral faces parallel to the VP. It is cut by a plane perpendicular to the VP and inclined at 30° to the HP. The plane meets the axis at a distance of 20 mm above the base. Draw the development of the lateral surfaces of the lower portion of the prism.
2. A hexagonal prism, edge of base 20 mm and axis 50 mm long, rests with its base on HP such that one of its rectangular faces is parallel to VP. It is cut by a plane perpendicular to VP, inclined at 45° to HP and passing through the right corner of the top face of the prism. (i) Draw the sectional top view. (ii) Develop the lateral surfaces of the truncated prism.
3. A pentagonal prism, side of base 25 mm and altitude 50 mm, rests on its base on the HP such that an edge of the base is parallel to VP and nearer to the observer. It is cut by a plane inclined at 45° to HP, perpendicular to VP and passing through the center of the axis. (i) Draw the development of the complete surfaces of the truncated prism.
4. A pentagonal prism of side of base 30 mm and altitude 60 mm stands on its base on HP such that a vertical face is parallel to VP and away from observer. It is cut by a plane perpendicular to VP, inclined at an angle of 50° to HP and passing through the axis 35 mm above the base. Draw the development of the lower portion of the prism.

Try : The exercise 8.1 and 8.2 in Solid Works and CREO software.

9. Exercise on Development of surfaces-2

9.1. Exercise on Development of surfaces of cylinder and cone

1. Draw the development of the lateral surface of the lower portion of a cylinder of diameter 50 mm and axis 70 mm when sectioned by a plane inclined at 40° to HP and perpendicular to VP and bisecting axis.
2. A Cone of base diameter 60 mm and height 70 mm is resting on its base on HP. It is cut by a plane perpendicular to VP and inclined at 30° to HP. The plane bisects the axis of the cone. Draw the development of its lateral surface.

9.2. Exercise on Development of surfaces of pyramid

1. Draw the development of the lateral surfaces of a square pyramid, side of base 25 mm and height 50 mm, resting with its base on HP and an edge of the base parallel to VP.
2. A square pyramid of base side 25 mm and altitude 50 mm rests on its base on the HP with two sides of the base parallel to the VP. It is cut by a plane bisecting the axis and inclined at 30° to the base. Draw the development of the lateral surfaces of the lower part of the cut pyramid.
3. A pentagonal pyramid side of base 30 mm and height 52 mm stands with its base on HP and an edge of the base is parallel to VP and nearer to it. It is cut by a plane perpendicular to VP, inclined at 40° to HP and passing through a point on the axis 32 mm above the base. Draw the sectional top view. Develop the lateral surface of the truncated pyramid.

Try: The exercise 9.1 and 9.2 in Solid Works and CREO software.

10. Orthographic views

Orthographic views are two-dimensional views of three-dimensional objects. Orthographic views are created by projecting a view of an object onto a plane which is usually positioned so that it is parallel to one of the planes of the object.

10.1. Exercise on Conversion of isometric view to orthographic projections using CAD

1. Draw the front view, side view and top view for the below Fig.10.1

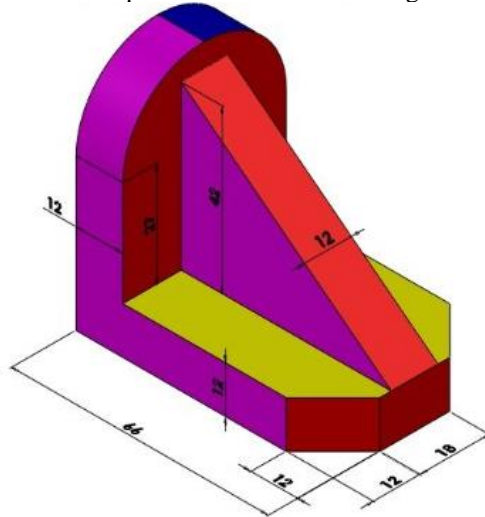


Fig.10.1

2. Draw the front view, side view and top view for the below Fig.10.2.

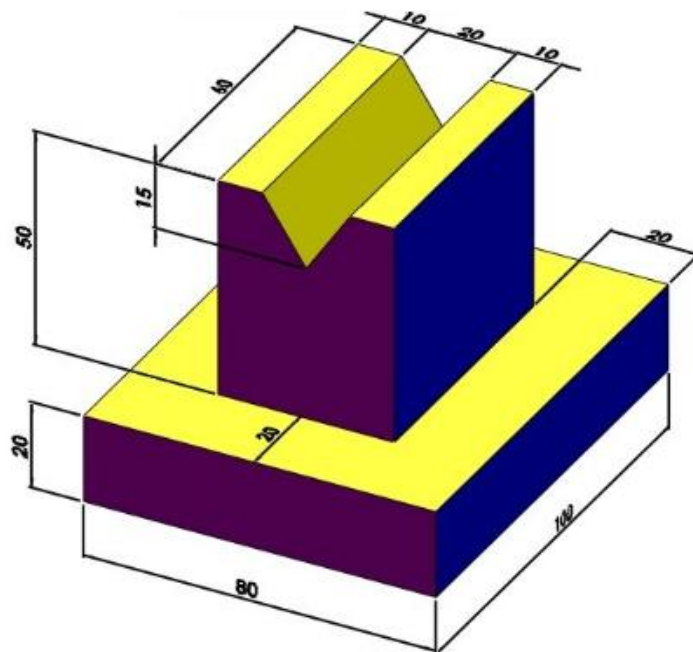
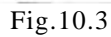


Fig.10.2

3. Draw the front view, side view and top view for the below Fig.10.3.



Isometric view of a mechanical part. The part has a base plate with a total length of 100 and a width of 80. On the left side, there is a semi-circular cutout with a radius of R20 and a depth of 15. The base plate has a thickness of 10. On the right side, there is a rectangular block with a width of 30 and a height of 15. The base plate has a thickness of 10. The total length of the part is 100.

Fig.10.4

Try : The exercise 11.2 in Solid Works and CREO software.

1. Isometric projections of solids

12.1. Exercise on conversion of orthographic view to isometric view using CAD

1. Draw the isometric view for the given orthographic views Fig.12.1

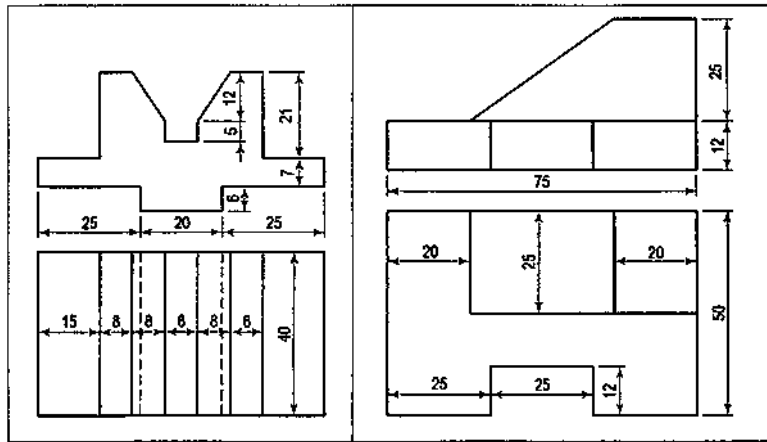


Fig.12.1

2. Draw the isometric view for the given orthographic views for Fig.12.2

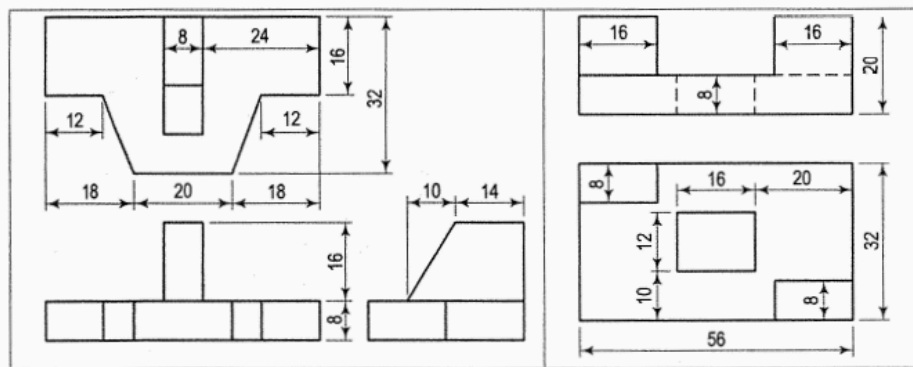


Fig.12.2

3. Draw the isometric view for the given orthographic views by assuming the dimensions for Fig.12.3

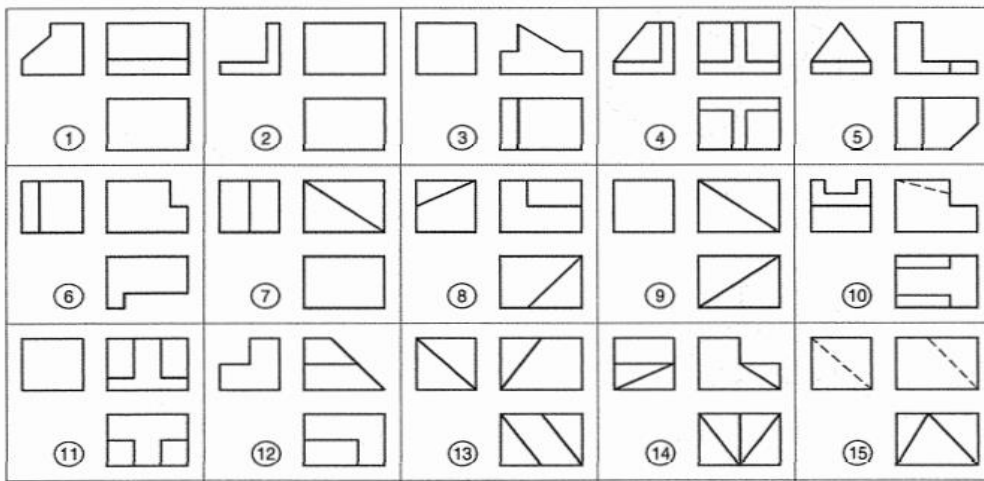


Fig.12.3

Try : The exercise 12 in Solid Works and CREO software.

13.Demonstration of SOLID WORKS Software

1. Introduction to SOLID WORKS

2. Demonstration of commands
3. 2D drawings
3. 3D drawings

14.Demonstration of CREO Software

1. Introduction to SOLID WORKS
2. Demonstration of commands
3. 2D drawings
3. 3D drawings

V. TEXT BOOKS:

1. Frederick E Giesecke, Alva Mitchell, Henry C Spencer, Ivan L Hill, John T Dygdon, James E. Novak, R. O. Loving, Shawna Lockhart, Cindy Johnson, *Technical Drawing with Engineering Graphics*, Pearson Education, 15th Edition, 2016.
2. Kulkarni D.M, Rastogi A.P. and Sarkar A.K., *Engineering Graphics with Auto CAD*. (Revised Edition), Prentice Hall India, New Delhi, 2011.
3. Donald Hearn, “*Computer Graphics*”, 12th Edition, Pearson, 2021.

VI. REFERENCE BOOKS:

1. Basant Agrawal and C M Agrawal, *Engineering Drawing*, McGraw Hill, 3rd Edition, 2018.
2. James M. Leake, Molly Hathaway Goldstein, Jacob L. Borgerson, *Engineering Design Graphics, Modelling and Visualization*, Wiley, 3rd Edition, 2020.

VII. ELECTRONIC RESOURCES:

1. <https://archive.nptel.ac.in/courses/112/103/112103019>.
2. <https://archive.nptel.ac.in/courses/112/105/112105294>.

VIII. MATERIALS ONLINE:

3. Course Template
4. Laboratory manual



INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal - 500 043, Hyderabad, Telangana

COURSE CONTENT

WEB AND MOBILE APPLICATIONS DEVELOPMENT								
II Semester: AE / ME / CE / ECE / EEE								
Course Code	Category	Hours / Week			Credits	Maximum Marks		
ACSD07	Skill	L	T	P	C	CIA	SEE	Total
		0	0	2	1	40	60	100
Contact Classes: Nil	Tutorial Classes: Nil	Practical Classes: 45			Total Classes: 45			
Prerequisite: There is no prerequisite required to this course								

I. COURSE OVERVIEW:

This course serves as foundation course on Web and Mobile applications development. It covers fundamental concepts to build modern web applications, the basics of HTML5 and CSS3 for Web application development. This environment will gain practical skills in mobile app development including user interface design, programming, and deployment.

II. COURSES OBJECTIVES:

The students will try to learn

- I. The characteristics, systematic methods, model for developing web applications.
- II. The concepts of client side programming with Bootstrap ,JavaScript, Ajax , Design user interfaces that follow best practices for usability and user experience
- III. The mobile application development for different platforms using appropriate tools and frameworks.
- IV. The user interface design with best practices for usability and user experience.

III. COURSE OUTCOMES:

At the end of the course students should be able to:

- CO1 Develop web pages using the HTML and CSS features with different layouts as per need of applications
- CO2 Use the JavaScript to develop the dynamic web pages.
- CO3 Use server-side scripting with PHP to generate the web pages dynamically using the database connectivity.
- CO4 Apply layout management and multi layout techniques to create adaptable user interface.
- CO5 Develop user interface for mobile application using widgets with event handling.
- CO6 Design pushes notifications for incoming messages.

IV. COURSE CONTENT:

1. Getting Started Exercises

1.1 Html layouts and links

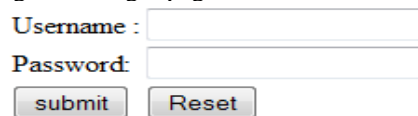
- Create a webpage with HTML describing your department use paragraph and list tags.
- Apply various colors to suitable distinguish key words, also apply font styling like italics, underline and two other fonts to words you find appropriate, also use header tags.
- Create links on the words e.g. “Wi-Fi” and “LAN” to link them to Wikipedia pages.

1.2 Web Application design formatting

- Develop a web application with background banner image and navigation menus.
- Develop a web application using left menu(list).
- Develop setting to change the theme of entire web Application.

1.3 Login page

Design the following static Login page.



Username :

Password :

1.4 Catalogue page

Design the following static web pages required for an online book store web site.

The catalogue page should contain the details of all the books available in the web site in a table.

- Branch
- Snap shot of Cover Page
- Author Name & Publisher.
- Price.
- Cart

1.5 Registration page

Design the following static web pages required for an online Registration from .

- E-mail id (Text field)
- First name (Text field)
- Last name (Text field)
- Phone number (text field)
- Gender (radio button)
- Date of birth (3 select boxes)
- Password (password filed)
- Retype the password (password filed)
- Submit query(button)
- Reset (button)

Exercises on java script

2.1 Validate page

Create a dynamic web page to validate the following fields.

- Name (Name should contains alphabets and the length should not be less than 6 characters).
- Password (Password should not be less than 6 characters length).
- E-mail id (should not contain any invalid and must follow the standard pattern name@domain.com)
- Phone number (Phone number should contain 10 digits only).

3. Online Recruitment System

Build a Online Recruitment System with the following functionalities:

1. **Administrator:** The information of the admin of the system is stored in this entity. It is stored data of login and the password. This provides security to the system and keeps the record of which user entered in the system at what instance of time.

It has the following attributes

- a. **Username:** It stores the name of the admin which acts as the unique name given to the manager of the firm.
 - b. **Password:** This attribute holds the secured keyword given to every manager of the educational institutions who need access to the system
 - c. **Login-Time:** The login time of the admin will be recorded in this field which helps in tracking the admin performance.
 - d. **Logout-Time:** It stores the logout time of the admin from the system
2. **Job Seeker:** To store the information of the candidate who registers to the system for participation in the recruitment process, the Job Seeker entity is created. It will have all the required data about the applicant.

It has following attributes

- a. **Can-Id:** Candidate Id is a unique number given to each candidate who registers in the system
 - b. **Can-Name:** Candidate Name is the information required for details of the applicant who registers in the system.
 - c. **Resume/CV:** In this attribute job, seekers can store their resumes which can be reviewed by the organization if they want to further check the candidate
 - d. **Projects:** Every candidate has many projects which they have created or worked on, which they want to share with them, recruiters.
 - e. **Personal-Info:** This attribute holds all the information the recruiters needed from the candidate apart from its technical skills and eligibility.
 - f. **Contact:** This stores the contact information of the candidate like phone number and email id
3. **Skill:** It is an essential part of the system as it works on both from recruiter and job seeker.

It has the following attributes

- a. **Skill-Id:** As every set of skills is unique so every set is given a unique number that acts as its id in the system.
 - b. **Technical:** It will store the technical skill by the applicant.
 - c. **Academic-details:** It will store the detailed information of the student in an academic field like high school marks, senior secondary marks. It will also have the graduation percentage is necessary for the skill.
 - d. **Co curricular-activities:** Every skill does not only require the technical talent of the student it also needs other social skills.
4. **Requisition:** In the recruitment management system, every employer will make the requisition of the application for the given post. From the application received from the number of the job seekers, the company will check their eligibility and decide whether he/she is compatible for the post or not. Then the shortlisted student will go for further rounds of the recruitment process as per the employer.

It has following attributes

- a. **Req-Id:** Every requisition present in the system given by each employer has unique numbers assigned to it.
 - b. **Emp-Id:** This attribute will hold the information of the employer who is giving this requisition.
 - c. **Post:** Every requisition given by the employer belongs to a specific post
 - d. **Skill-Id:** It is a reference to a skilled entity that implies the required skill set for the specific post.
 - e. **Package:** This attribute stores the information regarding the pay scale or salary
 - f. **Criteria:** In the criteria attribute, data regarding the process of recruitment is stored.
5. **Employer:** It will store the detailed information of the employer present or registered in the system. Thus, making it one of the essential entities in the system. Before applying for any post, the applicant also wants to know about the company or the employer who has given the requisition.

It has the following attributes

- a. **Emp-Id:** It is the unique number given to the employer to differentiate from others.
- b. **Emp-Name:** It will hold the name of the employer which is registered in the system.
- c. **Information:** This entity will hold the information of the employer like when it is founded, what kind of projects they are working on

- d. **Key-Persons:** It holds the data of the important persons of the company and the point of the contact for the interested candidates.
- 6. **Interview:** It has information about the interviews that will happen for the specific requisition. As the interview is an important part of any recruitment process as in this the employer and candidates come face to face and they can judge each other.

It has following attributes

- a. **Int-id:** It is the unique number given to each interview process that happens in the system
 - b. **Int-type-id:** As there are many kinds of interviews that can happen during the recruitment process.
 - c. **Req-Id:** It will hold the reference of the requisition for which a specific employer is conducting this interview.
 - d. **Remarks:** This has the remarks of the interviews which is given by the employers.
 - e. **Emp-Id:** It holds the data of the employer who is conducting this interview and managing the whole process.
7. **Interview-Type:** As there is various type of interview process which happens in the recruitment like technical for checking the technical knowledge of the candidate or the HR interview to test the social and interpersonal skill of the applicant.

It has following attributes

- a. **Int-type-id:** It is the unique number given to each type of interview
 - b. **Int-type-Name:** It holds the name of the type of interview.
8. **Offer-Letter:** When candidates apply for the job and it takes every criterion provided for a specific post. If it qualifies for the post the company will provide him/her with the offer letter. It is a document that makes the offer to the candidate for joining the employer's firm. It has detailed information regarding the package and the work environment of the company.

It has following attributes

- a. **Of-id:** Each offer letter is given a unique and distinct number
 - b. **Emp-Id:** It is the number given to the employer as it gives the information about which company has given this letter.
9. **Experience:** To show the talent each applicant writes the experience they have gain in the previous years. It helps the employer to see which applicant has the relevant knowledge required for the post. It is a weak entity dependent on the job seeker's profile. So, they can hire a person who has more potential than others.

It has following attributes

- a. **Exp-Details:** It holds the information candidate has learned during the work.
 - b. **Exp-Organization:** The name of the organization in which the applicant has gained the experience.
10. **Feedback:** A good system is where everyone has a say in everything. So, the feedback system is an entity that holds the feedback of the job seekers about their recruitment process for an employer or a requisition.

It has the following attributes:

- a. **Feed-Id:** It is a unique number given to each feedback. It makes the remark distinct and identifiable for the employers to take note of.
- b. **Emp-id:** It is a reference to the employer for which this feedback is given
- c. **Feed-details:** It holds the remark given by the applicant

See also for authentic understanding, click the link

<https://www.tracker-rms.com;>
[https://www.bullhorn.com/glossary/online-recruitment-system/;](https://www.bullhorn.com/glossary/online-recruitment-system/)
[https://www.manatal.com/;](https://www.manatal.com/)
<https://www.whatishumanresource.com/> and so on..
 Each student has to refer any one of the web sites stated above.

4. Student Counseling Management System

Build a Student Counseling Management System with the following functionalities

1. **Student:** This entity will store the record of the student registered for the counseling. It is essential for the system as we need a separate unit which has all the information regarding the student and their personal and professional data. So, it will have the 10th and 12th percentage, rank in the entrance exam will also be asked from student to give the preferences according to it.

It has the following attributes:

- a. **Name:** This will store the name of the student as it needed while saving the required
- b. **DOB:** The date of birth of the student as mention in the high school certificate and will act as the candidate key in the database model
- c. **Student-id:** It is a unique alphanumeric number assigned to every student who registers for the counseling
- d. **Password:** It is an alphanumeric field that has the constraint of having a minimum of 8 digits and at least one alphabet with numbers
- e. **10th Percentage:** This is used to get the student's percentage in high school which some universities check before giving admission to the student.
- f. **12th Percentage:** It is an academic field that requires the student to give information about the 12th standard.
- g. **Rank-Entrance-Exam:** This attribute gets the entrance exam rank from the student which they have got according to their performance in the exam

2. **Choices:** The entity is needed to get the preferences from the student based on their rank in the entrance exam. The number of choices given to the student can be multiple based on the institution that requires the seats available for the courses. These preferences will go to the courses entity which then checks which selection is eligible for the student, this evaluation will be based on the number of seats available and the rank of the student.

This is also will be referenced in the allotment of the seats where the allotted seats are recorded.

It has the following attribute:

- a. **Rank-Entrance-Exam:** This attribute gets the entrance exam rank from the student which they have got according to their performance in the exam.
- b. **1st Preference:** This will be a simple attribute that holds the first preference of the student which he/she will select from the pool of choices.
- c. **2nd Preference:** It will hold the second preference of the student as it is not always possible to get the first selected choice as it may have got filled or the student with a higher rank got that seat.

3. **Courses:** It will have all the courses provided by the institutes. All the courses will have institute code will differentiate if the same courses are provided by the different institution. Every Course has the define the seats available and the allotted seats along with the total seats available which will provide transparency in the system.

It has the following attribute:

- a. **Course-code:** A unique number given to every course not depends on which university to it belongs, it helps in maintaining the record unique and distinct so act as the primary key for the entity.
- b. **Institute-code:** Every institute that has register itself for the system of counseling has given a unique code that differentiates it from other institutes which have the same courses.
- c. **Total-seats :** It will have the total seats available for that course provided by the Institute.
- d. **No-of-seats:** It is a composite attribute which has the further information about a number of seats allotted and the vacant seat left for the course.

4. **Course-Name:** This entity will contain the name of the course and the unique id of that course. As the many institutes provide the same course it would be a difficult job to assign a different code to the same course. So, this field will be normalized and created a new entity out of it.

It has the following attribute:

- a. **Course-code:** A unique number is given to every unique course.
- b. **Course-name:** The name of the course will be stored in this attribute.

5. **Allotment of Seat:** This entity will store the information of the student after the seat is confirmed for allotment.

It has the following attribute:

- a. **Student-id:** Student unique id number.
 - b. **Rank-Entrance-exam:** Rank gave to the student as per his/her performance in the entrance exam
 - c. **Preference-no.:** The preference which got allotted, as per the choices are given to him.
 - d. **Course-code:** The course code which got selected for.
 - e. **Institute-code:** The institute allotted the seat.
6. **Institutes:** This entity will store the information regarding the institutes which are participating in the counseling. Each one of them has to publish their courses for which they want students with the number of the total seats.

It has the following attributes:

- a. **Institute-code:** Every institute that has register itself for the system of counseling has given a unique code that differentiates it from other institutes which have the same courses.
 - b. **Institute-name:** It will hold the name of the institute as mention by them in the registration process.
 - c. **City:** It depicts the city to which the institute belongs to as it helps as a candidate key.
 - d. **Password :** It is an alphanumeric field that has the constraint of having a minimum of 8 digits and at least one alphabet with numbers.
7. **Admitted-Student:** Once the student got allotted the seats and it gets confirmed by them Institute maintains their own record of which student has admitted to their institute and the courses what courses they have opted for.

It has the following attributes:

- a. **Student-id:** Student unique id number.
- b. **Course-code :** The course code which got selected for.
- c. **Institute-code:** The Institute for which the seat is allotted. Institute is related to this entity as every university will have a database of how many students have been admitted to its courses.

See also for authentic understanding, click the link

<https://collegedunia.com/exams/cuet/counseling>; <https://www.academiaerp.com/>;
<https://nchmcounseling.nic.in/>, https://www.cdac.in/index.aspx?id=edu_acts_Login_cv11 and
so on..

Each student has to refer any one of the web sites stated above..

5.Data Mart Management System

Build a Data Mart Management system with the following functionalities

1. Master maintenance
2. Shipment
3. Billing
4. Reports

1. Master Maintenance :

Data Mart Management System maintains the following master details for various purposes.

It has the following attributes

- a. Suppliers Details
- b. Sub location In-charges Details
- c. Retailers Details
- d. Products Details

2. Shipment :

Data Mart Management System provides you a number of functions to maintain the flow of goods in warehouse.

It has the following attributes

- a. Transfer inventory
- b. Put inventory on hold
- c. View inventory balances
- d. View inventory transactions

3. Billing :

Warehouse billing process allows the retailer to generate new bills and also allows

viewing of existing bills.

4. Reports :

The reports process allows the user to produce the reports necessary for day to day warehouse operations. The standard reports are ...

- a. Inventory Reports
- b. Purchase Order Reports
- c. Administrative Reports

See also for authentic understanding, click the link

<http://www.datamartsys.com/>; <https://netresultsgroup.com/>;
<https://www.indiamart.com/>; <https://uniondatamart.com/> and so on..
Each student has to refer any one of the web sites stated above.

6. Restaurant Reservation and Table Management Solutions

Restaurant Reservation and Table Management system is a Web application designed to help you (and your coworkers) to select the restaurant you are going to eat in. You can manage users, restaurants, menus, prices, give notations to each lunch, etc.

2. Reservation Management

- a. Easily enter or modify reservations while viewing guest histories.
- b. Capture phone numbers, email and mailing addresses.
- c. Allow management blocking and VIP pre-assignments.
- d. Reduce no-shows with enhanced customer tracking.
- e. Take reservations from your website or Open Table 24 hours.

3. Table Management

- a. Maximize seat utilization with walk-in and waitlist functionality.
- b. Instantly track covers for more efficient kitchen and server management. Increase table turns by tracking party status. Store multiple reservation sheets for holidays and special events.
- c. Hold and combine tables for large parties.
- d. Record and view shift notes for each day

2. Guest Management

- a. Identify regulars and VIPs
- b. Track customer preferences to meet and anticipate special requests
- c. View customer reservation histories at-a-glance
- d. Track special occasions such as guest birthdays and anniversaries Marketing Management
- e. Conduct powerful email marketing campaigns to increase repeat business.
- f. Print mailing labels to reach select target audiences.
- g. Track and reward concierge business.

3. Increase control

- a. Manage reservations from the back-office or any other location. Simultaneously control multiple restaurants from key centralized locations.
- b. Share guest data across sister restaurants.

See also for authentic understanding, click the link

<https://www.elluminatiinc.com/restaurant-reservation-system/>;
<https://restaurant.opentable.com/>; <https://www.tornosubitodubai.com/>;
<https://indiarestaurant.co.in/> and so on..
Each student has to refer any one of the web sites stated above

7. Secure Stock Exchange System using Web Services

This project implements a stock exchange is simply a system that is designed for the sale and purchase of securities of corporations and municipalities. A stock exchange sells and buys stocks, shares, and other such securities. In addition, the stock exchange sometimes buys and sells certificates representing commodities of trade.

The system “Secure Stock Exchange System using Web Services” consists of 3 modules

1. Stock Markets & Investments
2. Stock Options
3. Related Information

1. Stock Markets & Investments

- a. Stock Exchange Listing
- b. Stock Options & Analysis
- c. Stock Market Crash
- d. Selling Stock Certificates
- e. Stock Market Forecasts

2. Stock Options

Types of Stocks

- a. Stock Option Valuation
- b. Restricted Stock Options

3. Related Information

- a. Day Trading Stocks
- b. Stock Quotes & Stock Ticker
- c. Stock Charts
- d. Share Portfolio Management

See also for authentic understanding, click the link

<https://stocksandsecurities.adityabirlacapital.com/>; <https://www.nseindia.com/>;
<https://www.jpx.co.jp/english/>; <https://www.mstock.com/> and so on..

Each student has to refer any one of the web sites stated above.

8. Country Cargo and Express Couriers

This project is aimed at developing a data entry system that will enable the image assisted data entry for the cargo for the courier logistics. The scanned image if available will be displayed on the left side of screen when the Load ID is entered on the right side.

Module Description

1. The system should have appropriate login facility with relevant option like new user.
2. The system should allow admin to provide login id and passwords for the users with role
3. privilege option.
4. For appropriate user login the role privilege (Admin/Supervisor/User) should be selected.
5. The system should provide the Supervisor to set the priorities for the shipment load deliveries, also with services like ADD, UPDATE, VIEW, and DELETE on the shipment load item details.
6. Here the user is categorized as sender (source station) and receiver (destination station).
7. The system should provide the user (sender) to enter all his item details for shipment delivery.
8. The receiver after receiving the shipment delivery will be provided with the options of Viewing load details and also at the same time sends an act to both user (sender) and supervisor.

See also for authentic understanding, click the link

<https://worldwideexpresscouriers.com/> ; <https://www.vxpress.in/>;
<https://www.tciexpress.in/>; <https://www.bluedart.com/> and so on..

Each student has to refer any one of the web sites stated above

9. Food ordering application

Build a food ordering application with the following functionalities:

1. A 'Welcome page' which displays the logo and/or name of the app.
2. A 'Login Page' which asks for users' mobile number and password.
3. A 'Registration Page' which enables users to sign up for the app.
4. A 'Forgot Password Page' which enables users to reset their password.
5. A 'Navigation Drawer' with the app logo and user name on top and menu options to open the following pages:
 - a. Home
 - b. User Profile
 - c. Favourite Restaurants
 - d. Order History
 - e. Frequently Asked Questions (FAQs)
 - f. Log out
6. A 'My Profile' page (where the user's name, phone number, and address is displayed).
7. A 'Favorites' page (where the list of all favourite restaurants is displayed).
8. An 'Order History' page which lists the previously placed orders of the user.

9. An 'FAQ' page which lists some frequently asked questions. You can create any random questions which you feel could be relevant for a food delivery application. (Min. 5 questions)
10. A 'Logout' functionality which takes the user to the login page.
11. A 'Restaurant Details' page which displays the menu items of that particular restaurant, each item's price and the option to add an item to cart.
12. A 'Cart' page which lists the items added to cart and the total amount to be paid.

See also for authentic understanding, click the link

<https://www.swiggy.com>; <https://www.zomato.com/deliver-food>;

<https://www.ubereats.com>, <https://www.dineout.co.in> and so on..

Each student has to refer any one of the web sites stated above.

10. Music player application

Build a music player application with the following functionalities:

1. A 'Splash screen' (gradient background and app logo in center)
2. A 'Navigation drawer' with app logo section at the top along with links to 'All Songs', 'Favourites', 'Settings' and 'About Us'.
3. An 'All songs' screen (where of list all the tracks fetched from offline storage are displayed and user can sort the tracks by name or recently added). This will be the home screen of the app.
4. The app should be able to fetch and play .mp3 and .wav files.
5. A 'Favourites' screen (where list of all the favourite songs are displayed)
6. A 'Settings' screen (where the 'Shake to change song' feature can be enabled or disabled)
7. An 'About us' screen (where we will display information about the app developer and the app version)
8. A 'Now playing' screen with following features:
 - a. Track title and track artist
 - b. Play / Pause button
 - c. Next button
 - d. Previous button
 - e. Shuffle button
 - f. Loop button
 - g. Seek bar
 - h. Mark track as favourite or unfavourite it
 - i. Third party visualiser in upper half background
 - j. A 'Back to list' button in the header which should take the user to the screen he came from (kind of like back button behaviour).
 - k. Shake to change song
9. A 'Now playing' bar at the bottom with name of the track playing and play or pause feature. This would appear if the user has moved from 'Now playing' screen to 'All songs' screen or 'Favourites' screen without pausing the track.
10. Background play. The app will continue playing the track if the app gets closed (not killed) without the music being paused.
11. A notification saying "A track is playing in the background" only if the app gets closed (not killed) without the music being paused.

Primary color scheme: #9b2a58, #00032a

See also for authentic understanding, click the link

<https://open.spotify.com>; <https://www.hungama.com>;

<https://wynk.in/music> and so on..

Each student has to refer any one of the web sites stated above.

11. Smart Health Prediction

Build Android Smart Health Prediction application which can be used by all patients or their family members who need help in emergency with the following functionalities:

This application comprises of 3 major modules with their sub-modules:

1. A 'User page' with menu options to open the following pages:
 - a. A 'Patient Login page' which displays patient Login to the application using his ID and Password.

- b. A 'Patient Registration page' which asks if patient is a new user, he will enter his personal details and he will user Id and password through which he can login to the application.
 - c. A 'My Details page' which patient can view his personal details.
 - d. A 'Disease Prediction page' patient will specify the symptoms caused due to his illness. Application will ask certain question regarding his illness and application predict the disease based on the symptoms specified by the patient and application will also suggest doctors based on the disease.
 - e. A 'Search Doctor page' which the patient can search for doctor by specifying name, address or type.
 - f. A 'Feedback page' which the patient will give feedback this will be reported to the admin.
 - g. A 'Notification page' which the user needs to enter his/her prescription and then the user need to select a time on which he/she wants to receive notification. The application will send the notification regarding the prescribed medicine to the user on their device.
2. A 'Doctor page' with menu options to open the following pages:
 - a. A 'Doctor Login page' will access the application using his User ID and Password.
 - b. A 'Patient Details page' can view patient's personal details.
 - c. A 'Patient's Previous Details page' will get all information about patient's previous case history. That will help him to serve him better.
 3. A 'Admin page' with menu options to open the following pages:
 - a. A 'Admin Login page' can login to the application using his ID and Password.
 - b. A 'Add Doctor page' can add new doctor details into the database.
 - c. A 'Add Disease page' can add disease details along with symptoms and type.
 - d. A 'View Doctor page' can view various Doctors along with their personal details.
 - e. A 'View Disease page' can view various diseases details stored in database.
 - f. A 'View Patient page' can view various patient details who had accessed the application.
 - g. A 'View Feedback page' can view feedback provided by various users.

See also for authentic understanding, click the link

<https://open.spotify.com>; <https://www.hungama.com>;

<https://wynk.in/music> and so on..

Each student has to refer any one of the web sites stated above

12. Hostel Management Application

Develop the hostel management application which will help to manage the hostel. The hostel managers can keep track of the hostellers' in and out timings and their daily entries. This system is intended to help hostel admin by allowing them to save student records and information about their rooms. It helps the admin from the manual work from which it is very difficult to find the record of the students.

This application comprises of following functionalities:

1. A 'Admin page' with menu options to open the following pages:
 - a. A 'Manage Rooms' page which the admin can choose to add general rooms or bedrooms to the system. If the admin chooses a bedroom, they can add beds. They can add, update or delete rooms.
 - b. A 'Manage Students' page which the admin can view, add, update or delete students from the system. They can allocate rooms to the students. Also, they can view the attendance of a student.
 - c. A 'View Attendance' page which the admin can filter by date AND room OR student-id. They can view all the student's attendance.
2. A 'Student page' with menu options to open the following pages:
 - a. A 'Login' page which the student can log in to the system using a username and password.
 - b. A 'Profile' page which the student can add or update their profile details.
 - c. A 'Change Password' which they can also change their password to the new one.
 - d. A 'Home' page the student can scan the QR Code and a log will be added. They can view their room allocation details.
 - e. A 'View Attendance' page the students are able to view only their attendance, they can also filter it by date.
3. A 'Guardian page' which the ward wish to go out from campus for any purpose, the guardian will login through the registered login then the guardian will send the permission request to warden's android phone by specifying the purpose and also duration. After successful sent, the message is available in the warden login for further action.
4. A 'Warden page' after receiving the guardian's message, warden has a choice with options approve and reject. Based on the purpose of out campus, Warden Selects approve or reject. If approves then the message with the ward details are sent to security check which indicates the request made by the guardian is permitted.

See also for authentic understanding, click the link

<https://www.hostelsnap.com;>

[http://www.ifnoos.com/edu/college-university-software/hostel-management-system.aspx;](http://www.ifnoos.com/edu/college-university-software/hostel-management-system.aspx)

[https://www.ezeetchnosys.com/absolute;](https://www.ezeetchnosys.com/absolute) <https://www.resbird.com> and so on..

Each student has to refer any one of the web sites stated above

13. Stay safe women security application

Build stay safe women security project is used to provide highly reliable security system for the safety of women. The proposed system is based upon advanced sensors and GPS. The basic aim of the system is to develop a low cost solution for GPS based women tracking system (Women Safety System). The main objective of the system is to track the current location of the person which has an android enabled mobile by extracting the longitude and latitude of that target person.

This application comprises of following functionalities:

1. A 'Scream Alarm' page used perfect for the females as well as other users that need some kind of safety alarm in case they found out that someone is following or stalking them. It also consists of two other types of scream alarm. It's an initial distraction which will buy some time and allow the user to escape from the trouble.
 - a. Male voice scream
 - b. Police siren.

The user could select one of his/her choice from the "Settings" of the application, as keeping in mind the two other scream alarms are also added in this application as nowadays safety and security is everybody's concern.

2. A 'Fake Call Timer' page which the fake call timer allows the user to make fake calls in the time of need. It helps user to escape from an undesirable situation citing an important call from anyone who needs him/her urgently and rest depends upon user creativity. This feature also helps the user to escape from boring social events

In order to make a fake call the user have to select the "Fake Call" icon and after that user could write any name from which he/she wants a fake call. User could also set up the timer as per the requirement. The user could also set the default timer from the "Settings" icon of the application.

In a critical situation, the user just have to long term press the fake call button and automatically get a fake call as per the desired selected timer in the settings.

3. A 'Where Are You' which is used to find track friend. While first request is send by the sender. The sender will have to select the "Where Are You" icon and then a new dialog box of "Pick a Friend" will open up. The sender could select any friend and the request will be sent to the receiver. The receiver will accept that request from their end and a message will be sent to the receiver with the present location of the user.
4. A 'Track Me' which will track the user to view the exact dynamic location of the victim. First user have to send the Track Me request at the receivers end. The receiver will accept the request and then his/her name will appear on the friends you are tracking on the bottom of the application. The user could select that friend from there and then it will get automatically re-directed to the Google maps from where the user could view the exact location of the victim and also where's he/she heading to.
5. A 'Friends List' page which shows all the contact numbers of family and friends which are added by the user through contacts. This could be done by selecting the contact icon on the bottom right corner of the friends list.
6. A 'Settings' page which consists of the following features :-
 - a. A 'Emergency Services' page allows the Stay Safe Application to send emergency notifications and SMS with the exact location to the emergency contacts.
 - b. A 'Low Battery Alert' page alert feature allows the Stay Safe Application to send low battery alert and SMS to the emergency contacts.
 - c. A 'Set Scream Sound' page which the user could select any scream sound as per the requirement.
 - d. A 'Fake Call Timer (On Long press)' page which the user could set the fake call default timer as per the requirement.
7. A 'Emergency Distress Signal (SOS)' which the distress signal will be generated by the user in case of an emergency. In order to generate the distress signal the user have to shake up his/her phone, then a distress signal will appear at the user end with a default timer of 5 sec. In the end distress signal will be sent to the emergency contacts added by the user at the time of registration. The application send SMS and user details as well as the exact location of the user through a push notification at the receiver end, before sending a distress signal the user first have to turn on the emergency services from the settings of the application.

When user launches the application in his/her Android phone, the very first screen which lands is the Login Screen. First the user have to register himself by entering the details as the respective name and contact number of the user.

See also for authentic understanding, click the link

<https://womensafetywing.telangana.gov.in;>

<https://nirbhayaapp.com;> <https://safetipin.com;>

<https://wsww.smart24x7.com> and so on..

Each student has to refer any one of the web sites stated above

14. Controlling Anti Ragging Application

Controlling anti ragging system facilitate college students to register a criticism in opposition to ragging immediately. Students would be required to log in, after which they would be capable to register their complaints. The grievance will then be dispatched to the worried authorities for well-timed action and the motion will be initiated immediately. Previous archives reveal that well timed action was once taken in every case, which in flip resulted in a fall in such cases. Now, with the launch of the app, the Ministry hopes to wipe out the exercise completely.

This application comprises of following functionalities:

1. A 'User Module' page which the person can register with the machine then he/she will get the get entry to in the app. After login done if the any ragging is happened then without delay update the details. Every other flexibility is additionally on hand right here to contact the emergency wide variety also provided.
2. A 'College Admin' page which can operate the action on the failed complaints; complaints can be considered via the admin and can take the perfect action on the complaint.
 - a. A 'Add Complaint' page users can a make complaint against ragging. In this, those who are filling complaint as to fill some fields like Accused Name, Ragging Type, Mobile Number
 - b. A 'View Complaint' page which can view the complaints made by students.

See also for authentic understanding, click the link

<http://www.amanmovement.org;>

<https://www.nmc.org.in/ActivitiWebClient/open/initiateAntiRaggingHome;>

<https://www.antiragging.in;> <https://www.amanmovement.org/raggingmain.html> and so on..

Each student has to refer any one of the web sites stated above.

15. Extracurricular Event Tracking Application

Build Extracurricular activities are sources that have long been a part of the educational system; students participate in these activities, which are not part of the standard curriculum or teaching techniques. Participation in all of these activities, or even just one of them has been linked to social and academic success. Students who engage in extracurricular activities gain from the numerous options available to them. Having better grades, higher standardized test scores and educational attainment, attending school more consistently, and having a greater self-concept were all advantages of participating in extracurricular activities.

Our Extracurricular Event Tracking System is developed to track students' hours organizing and participating in cultural events and college fests. This system keeps students updated about upcoming events and maintains track of the events they have attended and the hours they have spent organizing various cultural programs. Thereby, encouraging participation.

This application comprises of following functionalities:

1. A 'Admin' page which contains the following features:
 - a. A 'Login' page which the admin can log in using their credentials.
 - b. A Manage Course which they can add, update, view and delete courses. The courses will be BSC/MSC/Diploma/Engineering, etc. They can add total hours.
 - c. A 'Manage Students' page which the admin can add, update, view and delete students. They can assign courses to the students.
 - d. A 'Manage Events' page which the admin can add, update, view and delete events. They can add details about a particular event. They can choose courses. They can add the link to the Meet.
 - e. A 'View Records' page which to view the records, the admin can choose by course. They can view the students' list and list of records.
2. A 'Student' page which contains the following features:
 - a. A 'Login' page which the students can log in using their credentials.
 - b. A 'Dashboard' page which the students can see the hours completed or the total hours. They can see up to 5 upcoming events ordered by date and time.
 - c. A 'Calendar' page the students can see the Day/Week/Month on the Calendar.
 - Day: The system will show today's events on the date. The user can also select any previous date to check any events that happened on that day.
 - Week: By selecting any week, the system will show one week's dates & events.

- Month: By selecting the month, the system will show the current month's dates and events.
- d. A 'Records' page which the system will show all the events attended.
- e. A 'Profile' page which the user can view their profile and change the password.
- f. A 'Forgot Password' page which if the user forgets their password, an OTP will be sent via Email or Phone number (Any 1). They would require to enter the correct OTP and reset the password.
- g. A 'Notification' page which the system will send a notification to the user before an event starts, and the dashboard checks the event unattended or hours not completed.

See also for authentic understanding, click the link

<http://www.amanmovement.org>;
<https://www.nmc.org.in/ActivitiWebClient/open/initiateAntiRaggingHome>;
<https://www.antiragging.in>; <https://www.amanmovement.org/raggingmain.html> and so on.
 Each student has to refer any one of the web sites stated above

16. Student management system

Build a student management system will have data on every student, and check the daily attendance of every student. The system comprises 3 major modules with their sub-modules as follows:

1. The 'Admin' page menu options to open the following pages

- a. A 'Login' page the admin can log in to the system using a username and password.
- b. A 'Manage Students' page the admin can manage student details and they can add, update and delete details. Also, add a fingerprint while adding the student's details.
- c. A 'Manage Teacher' page admin will also add teacher's details and add, update and delete details. Teachers will be assigned subjects.
- d. A 'Manage Grades' page can manage students' grades and add, update or delete grades.
- e. A 'Manage Subjects' page can also manage subjects by adding, updating and deleting subjects.
- f. A 'Manage Appointments' page applies to meet parent and waits for a reply.
- g. A 'Appointments' page parents applied to meet the teacher/admin. The admin will respond to the parent's appointment request.
- h. A 'Manage Leaves' page able to manage pending and past student leave. They can view a list of all the applied student's leaves. The admin can view all the pending leaves applied by parents. The admin will approve or reject the leave application. Leave Note: They will be able to send a note to the parent related to leaves.
- i. A 'Medical Certificate' page will be able to view the medical certificate submitted by parents. They can send a note with an acknowledgement in response.
- j. A 'View Complaint' page can view pending and replied complaints. They will be able to respond to the complaints.
- k. A 'Warning Letters' page contains the admin can add and view a list of all students with attendance below 80% / total course work below 60%. They can send a warning letter to their parents. Admin will be able to view all warning letters and replies by parent.
- l. A 'Attendance' page contains Entry and Exit Lecture-wise Attendance. See the attendance list of all Students grade-wise against 2 dates.
- m. A 'Reports' page contains can view reports of all the student's attendance lists, grade-wise. They can view students' attendance reports for 2,4,6-months with lecture, entry and exit details. They can also view the academic reports of all the students.
- n. A 'Entry/Exit' page can add entry and exit of the student biometric.

2. A 'PARENTS' page contains following features

- a. A 'Login' page can log in to the system using login credentials.
- b. A 'Profile' page can add or update details to their profile.
- c. A 'Change Password' can also change their old password to the new one.
- d. A 'Forget Password' used if they forget their login password, they can receive a link to change the password on their registered email-id.
- e. A 'Appointments' page contains the parent can apply to meet the admin/teacher and waits for their response. The parent will also receive the admin's request to meet them and they can respond to the request.
- f. A 'Leaves' page contains Parents can view all the leaves including pending leaves. They can view the list of all their applied leaves. Parents can view leaves that have not been replied to by an admin. They can also cancel their applied leaves.
- g. A 'Medical Certificate' page contains Add: Parents can upload the medical reports of their child while applying for leave. They can view all the previously uploaded lists and replies from the admin.

- h. A 'Complaint' page contains Parents can add a new complaint. They can view a list of all the complaints and replies from the admin.
 - i. A 'Warnings' page list of the warning letters sent by admin and parents can respond to them.
 - j. A 'Attendance' page can view their child's attendance, lecture-wise. Also, they can view the attendance list of the child against 2 dates.
 - k. A 'Reports' page contains following features: Academic attendance reports can be viewed by the admin. Parents can view attendance reports of 2/4/6 Months with lecture, entry and exit details. They can also view their child's academic report.
 - l. A 'Notifications' page will receive notifications such as when appointment status changes or admin applied, leaves approved/rejected, medical certificate reply, complaint reply, warning letter, student entry/exit, academic marks uploaded by teacher.
3. A 'TEACHER' page contains following features
- a. A 'Login' page can log in to the system using a username and password.
 - b. A 'Profile' page can add or update details to their profile.
 - c. A 'Change Password' can also change their passwords.
 - d. A 'Forget Password' if they forget their login password, they can receive a link to change the password on their registered email-id.
 - e. A 'My Subjects' page contains following features
Assigned Subjects: A list of all assigned subjects can be viewed grade-wise.
Students List: The teacher can view the list of the students in that Grade.
 - m. A 'Attendance' page contains they can select grade and subject to view attendance. Biometric Attendance: They can take the biometric attendance of students. See the Attendance list of all Students grade-wise against 2 dates, lecture-wise.
 - n. A 'Academics' page contains teachers can choose the grade, subject and student for their academic reports. They can view, add and update students' academic details. Teachers can add marks on tests, assignments and final exams. They can update existing marks on tests, assignments and final exams.
 - o. A 'Complaint' page contains teachers can add new student complaint. They can view list of all the complaints and replies from admin.

See also for authentic understanding, click the link

<https://www.itms.co.in/products/student-information-system-sis>,

<https://samvidha.iare.ac.in>;

<https://themeforest.net/item/clever-course-learning-management-system-theme/8645312>;

<https://markerspro.in> and so on.

Each student has to refer any one of the web sites stated above

17. Pharm easy application

Build a pharm easy application for ordering medicines and tests. This application comprises of following functionalities

1. A "Home Page" which consists of a list medicines & health care product.
2. A "Lab Tests page" contains different medical tests.
3. A "Health Care Products page" contains vitamins and necessary to our body.
4. A "Offers Page" offers on different products and coupons.
5. A "Account Page" contains details of individual customer.
6. A "Products Page" contains a list of products.
7. A "Detailed View of Product" detailed view of selected product.
8. A "Cart page" list of items we selected to buy.
9. A "Delivery Details Page" consists of customer address and contact details.
10. A "Payment Page" how to make payments (cod or online payments).
11. A "Success page" show transaction success.
12. A "Bottom APP Bar" consists of logos which opens the corresponding pages(fragments).
 - a. Home page
 - b. Test Page
 - c. Health Care Page
 - d. Offers Page
 - e. Account Page

See also for authentic understanding, click the link

<https://pharmeasy.in>; <https://www.netmeds.com>;

<https://www.medlife.com>; <https://www.practo.com> and so on.

Each student has to refer any one of the web sites stated above.

18. News Application

Build a news application is to connect news articles from all around the world and deliver it to user as fast as possible in best visualize way. This application comprises of following functionalities:

1. A 'User Interface' page should be able to select from different categories, countries and newspaper. Short News as list view with header, little description and image before showing full article can be helpful to user to determine what type of news they are looking for. View Holder can be used for this list view for better and fast experience.
2. A 'Admin Panel' page controls the User and Writers logins from database. Writers can add news, update and delete from its database as per required. Main Admin can add Users, Writers, and News. He can also approve, update and delete it. Using this approach, we can create network in local areas connect by writers and local admins which will provide news at local level and we can also implement location feature which will update local news of different location or city.
3. A 'Global Support' page contains different type of newspaper will be available from all around the world in different languages with this user will be able to get news from all around the world.
4. A 'Short News' page News will be displayed in short format with title, image and little description in list view. It will help user to access required news faster.
5. A 'Search Option' page which user will be able to search from not only one source but many different sources available within API.
6. A 'Favorites / Offline Reading' page which News can be added as favorites which will automatically will be saved for offline reading.
7. A 'Sharing' page which user will be able to share news easily on social media.

See also for authentic understanding, click the link

<https://indianexpress.com>; <https://www.bbc.com/news>;

<https://www.ndtv.com>; <https://www.hindustantimes.com> and so on..

Each student has to refer any one of the web sites stated above.

V. TEXT BOOKS:

1. Thomas A. Powell, "*The Complete Reference*", "HTML and CSS", 5th Edition, 2017
2. Elisabeth Robson , Eric Freeman," *Head First HTML and CSS: A Learner's Guide to Creating Standards-Based Web Pages*", 2nd Edition, 2012
3. Adam Boduchand Roy Derks, "*React and React Native: A Complete Hands-on Guide to Modern Web and Mobile Development with React.js*", 3rd Edition, 2020.
4. RetoMeier, "*ProfessionalAndroid 4 Application Development*", 1st Edition, Wile Publication

VI. REFERENCE BOOKS:

1. W Hans Bergsten, "*Java Server Pages*", O'Reilly, 3rd Edition, 2003.
2. D. Flanagan, "*Java Script*", O'Reilly, 6th Edition, 2011.
3. Jon Duckett, "*Beginning Web Programming*", WROX, 2nd Edition, 2008.
4. Bill Phillips and Chris Stewart, "*Android Programming*", The Big Nerd Ranch Guide, 3rd Edition, 2017.
5. Dawn Griffiths, David Griffiths, "*Head First Android Development: A Brain-Friendly Guide*", 2017.
6. Antonio Leiva , Kotlin for Android Developers: Learn Kotlin while developing an Android App, Create Space Independent Publishing, 2016

VII. ELECTRONIC RESOURCES:

1. <https://www.codecademy.com/learn/paths/web-development/>
2. <https://nptel.ac.in/courses/106/105/106105084/>
3. <https://medium.com/@aureliomerenda/create-a-native-web-app-with-react-native-web-419acac86b82>
4. <https://www.coursera.org/learn/react-native>
5. <https://desirecourse.net/react-native-and-redux-course-using-hooks>
6. <https://www.javatpoint.com/android-tutorial>
7. <https://www.tutorialspoint.com/android/index.htm>
8. <https://docs.flutter.dev/>
9. <https://developer.android.com/courses/android-basics-compose/course>
10. <https://developer.android.com/guide>

VIII. MATERIALS ONLINE

1. Course template

2. Lab Manual



INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal - 500 043, Hyderabad, Telangana

COURSE CONTENT

GENDER SENSITIZATION								
II Semester: AE / ME / CE / ECE / EEE / CSE / CSE (CS) / CSE(DS) / CSE (AI & ML) / IT								
Course Code	Category	Hours / Week			Credits	Maximum Marks		
AHSD10	Mandatory	L	T	P	C	CIA	SEE	Total
		-	-	-	-	-	-	-
Contact Classes: Nil	Tutorial Classes: Nil	Practical Classes: Nil			Total Classes: Nil			
Prerequisite:								

I. COURSE OVERVIEW:

The course aims at raising awareness of gender equality among students from sociological, cultural, psychological, legal and economical perspectives, thereby empowering them to communicate better in a cross-cultural work ambience. At the end of this course the students expose to better egalitarian interactions between men and women and to enable them see diversity and inclusiveness as assets in a globalized scenario.

II. COURSES OBJECTIVES:

The students will try to learn

- The basic gender concepts and their application to the long- term implementation of programming and initiatives.
- The gender roles, expectations and issues and their impact on as the community's day-to-day life.
- The more egalitarian interactions between men and women.

III. COURSE OUTCOMES:

At the end of the course students should be able to:

- CO 1 Develop a better understanding of important issues related to gender in contemporary India.
- CO 2 Sensitize to the basic dimensions of the biological, sociological, psychological and legal aspects of gender.
- CO 3 Acquire insight into the gendered division of labor and its relation to politics and economics.
- CO 4 Attain a finer grasp of how gender discrimination works in our society and how to counter it.
- CO 5 Men, women and professionals will be better equipped to work and live together as equals.
- CO 6 Know the new laws that provide protection and relief to women.

IV. SYLLABUS:

MODULE-I: UNDERSTANDING GENDER

Introduction: definition of gender, basic gender concepts and terminology, exploring attitudes towards gender, construction of gender. Socialization: making women, making men, preparing for womanhood. growing up male, first lessons in caste.

MODULE – II: GENDER ROLES AND RELATIONS

Two or many; struggles with discrimination; Gender roles and relations: types of gender roles, gender roles and relationships matrix, missing women, sex selection and its consequences, declining sex ratio, demographic consequences; Gender Spectrum: Beyond the Binary.

MODULE-III: GENDER AND LABOUR

Division and valuation of labour; Housework: the invisible labor, my mother doesn't work. Share the load work: its politics and economics, fact and fiction. unrecognized and unaccounted work; gender development issues gender, governance and sustainable development; gender and human rights; gender and mainstreaming.

MODULE-IV: GENDER AND BASED VIOLENCE

The concept of violence; types of gender; based violence, gender, based violence from a human Rights perspective, Sexual harassment: say no sexual harassment, not eve-teasing-, coping with everyday harassment, further reading, chupulu. Domestic violence: speaking out is home a safe place, when women unite (Film). rebuilding lives, thinking about sexual violence blaming the victim, i fought for my Life.

MODULE-V: GENDER AND CULTURE

Gender and film; gender and electronic media; gender and advertisement; gender and popular, literature, Gender

development issues, gender issues, gender sensitive language, gender and popular literature. Just relationships: being together as equals; Mary kom and onler, love and acid just do not mix, love letters, mothers and fathers, Rosa Parks, the brave heart.

V. TEXT BOOKS:

1. A. Suneetha, Uma Bhargubanda, Duggirala Vasanta, Rama Melkote, Vasudha Nagaraj, Asma Rasheed, Gogu Shyamala, Deepa Sreenivas and Susie Tharu The Textbook, *Towards a World of Equal: A Bilingual Textbook on X Gender*, published by Telugu Academy, Telangana Government, 1st Edition, 2015.

VI. REFERENCE BOOKS:

1. Kadambari V, *Gender Studies: A Primer*. Rajiv Gandhi National Institute of Youth Development, Sriperumbudur. 1st Edition, 2009.
2. C. Rajya Lakshmi Kalyani, D.S. Vittal, A. Kanaka Lakshmi, P. Chandrakala, B. Lavanya., *Gender Sensitization*, Himalaya Publishing House. 1st Edition, 2017.

VII. ELECTRONIC RESOURCES:

1. <https://en.unesco.org/women-make-the-news-2017/resources>
2. http://ncw.nic.in/sites/default/files/Booklet-%20Gender%20Sensitization_0.pdf

COURSE CONTENT

ANALYSIS OF ELECTRICAL NETWORKS								
III Semester: EEE								
Course Code	Category	Hours / Week			Credits	Maximum Marks		
AEED05	Core	L	T	P	C	CIA	SEE	Total
		3	0	0	3	40	60	100
Contact Classes: 48	Tutorial Classes: Nil	Practical Classes: Nil			Total Classes: 48			

I. COURSE OVERVIEW:

This course introduces the basic concepts of network theory which is the foundation for all subjects of the electrical engineering discipline. The emphasis of this course is laid on the basic analysis of circuits which includes three phase circuits, transient analysis of DC and AC circuits, network functions, and two port network parameters, Fourier analysis of AC circuits, design and analysis of filters.

II. COURSE OBJECTIVES:

The students will try to learn:

- The three phase systems for star and delta connected systems and perform three phase power calculations for balanced and unbalanced loads.
- The necessary mathematical background for the transient analysis of DC and AC circuits and study the transients using differential equation and Laplace transform approach for series and parallel circuits
- The concept of locus diagram for series and parallel circuits and discuss network functions and the stability criteria
- The steady state response of complex electrical circuits with AC supply and application of concept of electrical resonance
- Classify and design different types of filters and study their characteristics.

III. COURSE OUTCOMES:

After successful completion of the course, students should be able to:

- CO1** Outline the steady state response and resonance of electrical circuits to AC supply
- CO2** Understand the concept of initial conditions of RLC elements to determine the transient response of first and second order electric circuits using differential equation approach and Laplace transform technique.
- CO3** Illustrate the locus diagram for series and parallel circuits and describe the network functions in time domain and frequency domain approach.
- CO4** Solve the relation between line and phase quantities of three phase star and delta connected systems to analyze balanced and unbalanced circuits.
- CO5** Demonstrate the operation of wattmeter to measure the three-phase active and reactive power in three phase systems.
- CO6** Develop the various types of active filters and understand their characteristics, execute digital simulation using MATLAB.

IV. COURSE CONTENT:

MODULE-I: ANALYSIS OF AC CIRCUITS (10)

Steady state analysis: Steady state analysis of RL, RC and RLC circuits (in series, parallel and series parallel combinations) with sinusoidal excitation; **Resonance:** Series and parallel resonance, concept of band width and Q factor.

MODULE-II: SOLUTION OF FIRST AND SECOND ORDER NETWORKS (09)

Transient response: Initial conditions, transient response of RL, RC and RLC series and parallel circuits with DC and AC excitations, differential equation and Laplace transform approach.

MODULE-III: LOCUS DIAGRAMS AND NETWORKS FUNCTIONS (10)

Locus diagrams: Locus diagrams of RL, RC, RLC circuits.

Network Functions: The concept of complex frequency, physical interpretation, transform impedance, series and parallel combination of elements, terminal ports, network functions for one port and two port networks, poles and zeros of network functions, significance of poles and zeros, properties of driving point functions and transfer functions, necessary conditions for driving point functions and transfer functions, time domain response from pole-zero plot.

MODULE-IV: THREE PHASE CIRCUITS (10)

Three phase circuits: Star and delta connections, phase sequence, relation between line and phase voltages and currents in balanced systems(both Y& Δ), three phase three wire and three phase four wire systems, analysis of balanced and unbalanced three phase circuits, measurement of active and reactive power.

MODULE-V: FILTERS (09)

Filters: Classification of filters, filter networks, classification of pass band and stop band, characteristic impedance in the pass and stop bands, constant-k low pass filter, high pass filter, m-derived T-section, band pass filter and band elimination filter.

V. TEXT BOOKS:

1. A Chakrabarthy, "Electric Circuits", Dhanpat Rai & Sons, 6th edition, 2010.
2. A Sudhakar, Shyamamohan S Palli, "Circuits and Networks", Tata McGraw Hill, 4th edition, 2010.

VI. REFERENCE BOOKS:

1. John Bird, "Electrical Circuit Theory and technology", Newnes, 2nd edition, 2003.
2. C L Wadhwa, "Electrical Circuit Analysis including Passive Network Synthesis", New Age International, 2nd edition, 2009.
3. David A Bell, "Electric Circuits", Oxford University press, 7th edition, 2009.
4. M E Van Valkenberg, "Network Analysis", Prentice Hall India, 3rd edition, 2014.
5. Rudrapratap, "Getting started with MATLAB: A Quick Introduction for Scientists and Engineers", Oxford University Press, 1st edition, 19994.

VII. ELECTRONIC RESOURCES:

- 1 <https://www.igniteengineers.com>
- 2 <https://www.ishuchita.com/PDF/Matlab%20rudrapratap.pdf>
- 3 <https://www.ocw.nthu.edu.tw>
- 4 <https://www.uotechnology.edu.iq>

VIII. MATERIALS ONLINE:

1. Course template
2. Tutorial question bank
3. Tech talk topics
4. Open end experiments
5. Definitions and terminology
6. Assignments
7. Model question paper - I
8. Model question paper - II
9. Lecture notes
10. E-learning readiness videos (ELRV)
11. Power point presentation



INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal - 500 043, Hyderabad, Telangana

COURSE CONTENT

DC MACHINES AND TRANSFORMERS								
III Semester: EEE								
Course Code	Category	Hours / Week			Credits	Maximum Marks		
AEED06	Core	L	T	P	C	CIA	SEE	Total
		3	0	0	3	40	60	100
Contact Classes: 48	Tutorial Classes: Nil	Practical Classes: Nil			Total Classes: 48			
Prerequisite: Electrical Circuits, Engineering Physics								

I. COURSE OVERVIEW:

This course deals with the basic theory, construction, operation, performance characteristics and application of electromechanical energy conversion devices such as DC generators and motors. It also gives an in-depth knowledge on the operation of single phase and three phase transformers and its testing. It also focus on the auto transformers, on-load, off-load tap changers which are widely used in real time applications.

II. COURSE OBJECTIVES:

The students will try to learn:

- The principles of single excited and multiple excited systems leading to the energy balance equations.
- The construction, working and operation of self and separately excited DC machines.
- The performance characteristics of different DC machines when they are under no load and load conditions.
- The energy transformation using single and poly phase transformers under no load and load conditions.

III. COURSE OUTCOMES:

After successful completion of the course, students should be able to:

- CO1** Describe the load sharing capabilities and reliability of electrical machines using parallel operation under various loading conditions.
- CO2** Use the concepts of complex algebra, phasor operations, principles of electromagnetism and circuit theory. for analyzing the performance related issues in electrical machines.
- CO3** Demonstrate the working of linear machine as generator and motor by applying electromagnetic laws and its mathematical models under different loading conditions.
- CO4** Identify various control strategies for calculating the performance parameters and voltage regulation of electrical machines.
- CO5** Illustrate the equivalent circuits and connections of single and three phase transformers and auto transformers for power system analysis.

IV. COURSE CONTENT:

MODULE-I: DC GENERATORS (10)

DC generators: Principle of operation, construction, lap and wave windings, simplex and multiplex windings, commutator, EMF equation, types of DC generators, Armature reaction: Cross magnetization and demagnetization, ampere turns per pole, compensating winding; Commutation: Methods of improving commutation; Open circuit characteristics, voltage buildup, critical field resistance and critical speed, causes for failure to self-excite and remedial measures, load characteristics of shunt, series and compound generators; Conditions and necessity for parallel operation, load sharing, equalizer bars, cross connection of field windings, numerical problems.

MODULE-II: DC MOTORS (09)

DC motors: Principle of operation, back EMF, torque equation, types of DC motors, condition for maximum power developed, armature reaction and commutation, characteristics, types of starters, numerical problems.

MODULE-III: PERFORMANCE OF DC MACHINES (10)

Losses and efficiency: Types of losses, efficiency, condition for maximum efficiency Speed Control Methods: Speed control of DC machines; Testing methods: Swinburne's test, brake test, retardation test, separation of stray losses, Hopkinson's test, and field's test, numerical problems.

MODULE-IV: SINGLE PHASE TRANSFORMERS (10)

Single phase transformers: Principle of operation, construction, types of transformers, EMF equation, concept of leakage flux and leakage reactance, operation of transformer under no-load and on-load, phasor diagrams, equivalent circuit, efficiency, regulation and all day efficiency; Cooling methods; Testing of transformers: objectives, polarity test, measurement of resistance, OC and SC tests, back to back test, heat run test, parallel operation, numerical problems.

MODULE-V: POLY PHASE TRANSFORMERS (09)

Three phase transformers: Principle of operation, star to star, delta to delta, star to delta, delta to star, three phase to six phase, open delta connection, Scott connection; Auto transformers: Principles of operation, equivalent circuit, merits and demerits, no load and on load tap changers, harmonic reduction in phase voltages, numerical problems.

V. TEXT BOOKS:

1. P S Bimbhra, "Electrical Machinery", Khanna Publishers, 1st edition, 2011.
2. I J Nagrath and D P Kothari, "Electric Machines", McGraw Hill Education, 1st edition, 2010.
3. J B Guptha "Theory and performance of Electrical machines", S.K.Kataria & Sons Publishers 14th edition, 2009.

VI. REFERENCE BOOKS:

1. M G Say, E O Taylor, "Direct Current Machines", Longman Higher Education, 1st edition, 1985.
2. M G Say, "Performance and design of AC machines", CBS Publishers, 1st edition, 2002.
3. A E Fitzgerald and C Kingsley, "Electric Machinery", New York, McGraw Hill Education, 1st edition, 2013.
4. M V Deshpande, "Electrical Machines", PHI Learning Private Limited, 3rd edition, 2011.

VII. ELECTRONIC RESOURCES:

1. <https://www.electrical4u.com>
2. <https://www.freevideolectures.com>

VIII. MATERIALS ONLINE:

1. Course template
2. Tutorial question bank
3. Tech talk topics
4. Open end experiments
5. Definitions and terminology
6. Assignments
7. Model question paper - I
8. Model question paper - II
9. Lecture notes
10. E-learning readiness videos (ELRV)
11. Power point presentation



INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal - 500 043, Hyderabad, Telangana

COURSE CONTENT

ELECTROMAGNETIC FIELDS								
III Semester: EEE								
Course Code	Category	Hours / Week			Credits	Maximum Marks		
AEED07	Core	L	T	P	C	CIA	SEE	Total
		3	0	0	3	40	60	100
Contact Classes: 48	Tutorial Classes: -	Practical Classes: Nil			Total Classes: 48			
Prerequisite: Electrical Circuits								

I. COURSE OVERVIEW:

This course will equip the students with good understanding of underlying principles and laws in electromagnetic fields and waves. The concepts of vector algebra, principles and basic laws of electrostatics, characteristics and properties of conductors and dielectrics, behavior of static magnetic field and application of Ampere's law, determination of force in magnetic field and magnetic potential, concept of time varying fields and propagation of electro-magnetic waves.

II. COURSE OBJECTIVES:

The students will try to learn:

- The behavior of charge under rest with static electric field in terms of electric field intensity, electric displacement and electric potential
- The charge distribution in conductors, dielectrics and condensers.
- The sources to develop constant and variable magnetic field to study effect of these fields in terms of magnetic field intensity, magnetic displacement, and magnetic potential
- The nature of electromagnetic wave propagation in free space, conductors, and dielectric materials

III. COURSE OUTCOMES:

- CO1** Make use of Vector Calculus, Coulomb's Law and Gauss Law for obtaining electric field intensity, Potential and behavior of electrostatic field
- CO2** Calculate the capacitance of different physical configuration based on the behavior of the conductors and dielectric materials
- CO3** Demonstrate Biot-Savart law and Ampere circuital law for derivation of magnetic field intensity due to different current carrying conductors.
- CO4** Predict the force due to moving charge/current in the static magnetic field, thereby obtaining the inductance for different configurations of wires and energy stored in the coil
- CO5** Apply the Faraday's law of Electromagnetic induction and Maxwell Equations to produce a wave equation for the free- space, insulators and conductors for propagation of electromagnetic waves

III. COURSE CONTENT:

MODULE-I: ELECTROSTATICS (10)

Introduction to Cartesian, cylindrical and spherical co-ordinates. Conversion of one type of co-ordinates to another; Electrostatic fields: Coulomb's law, electric field intensity due to line and surface charges, work done in moving a point charge in an electrostatic field, electric potential, properties of potential function, potential gradient, Gauss's law, application of Gauss's law, Maxwell's first law, Laplace's and Poisson's equations, solution of Laplace's equation in one variable.

MODULE-II: CONDUCTORS AND DIELECTRICS (08)

Dipole moment, potential and electric field intensity due to an electric dipole, torque on an electric dipole in an electric field, behavior of conductors in an electric field, electric field inside a dielectric material, polarization, conductor and dielectric, dielectric boundary conditions, capacitance of parallel plate and spherical and coaxial capacitors with composite dielectrics, energy stored and energy density in a static electric field, current density, conduction and convection current densities, Ohm's law in point form, equation of continuity.

MODULE-III: MAGNETOSTATIC (10)

Biot-Savart's law, magnetic field intensity, magnetic field intensity due to a straight current carrying filament, magnetic field intensity due to circular, square and solenoid current carrying wire, relation between magnetic flux, magnetic flux density and magnetic field intensity, Maxwell's second equation, $\text{div}(\mathbf{B})=0$. Magnetic field intensity due to an infinite sheet of current and a long current carrying filament, point form of Ampere's circuital law, Maxwell's third equation, $\text{Curl}(\mathbf{H})=\mathbf{J}_c$, field due to a circular loop, rectangular and square loops

MODULE-IV: FORCE IN MAGNETIC FIELD AND MAGNETIC POTENTIAL (10)

Moving charges in a magnetic field, Lorentz force equation, force on a current element in a magnetic field, force on a straight and a long current carrying conductor in a magnetic field, force between two straight long and parallel current carrying conductors, magnetic dipole and dipole moment, a differential current loop as a magnetic dipole, torque on a current loop placed in a magnetic field; Vector magnetic potential and its properties, vector magnetic potential due to simple configurations, Poisson's equations, self and mutual inductance, Neumann's formula, determination of self-inductance of a solenoid, toroid and determination of mutual inductance between a straight long wire and a square loop of wire in the same plane, energy stored and density in a magnetic field, characteristics and applications of permanent magnets.

MODULE-V: TIME VARYING FIELDS AND FINITE ELEMENT METHOD (09)

Faraday's laws of electromagnetic induction, integral and point forms, Maxwell's fourth equation, $\text{curl}(\mathbf{E})=\partial\mathbf{B}/\partial t$, statically and dynamically induced EMFs, modification of Maxwell's equations for time varying fields, displacement current. Derivation of Wave Equation, Uniform Plane Waves, Maxwell's equation in phasor form, Wave equation in Phasor form, Plane waves in free space and in a homogenous material. Wave equation for a conducting medium, Plane waves in loss dielectrics, Propagation in good conductors, Skin effect. Poynting theorem

IV. TEXT BOOKS:

1. K.B. MadhuSahu, "Electromagnetic Fields", Scitech Ltd., 2nd edition, 2014.
2. David J Griffiths, "Introduction to Electrodynamics" Pearson Education Ltd., 4th edition, 2014.
3. Sunil Bhooshan, "Fundamentals of Engineering Electromagnetics", Oxford University Press, 1st edition, 2012.
4. E Kuffel, W S Zaengl, J Kuffel, "High Voltage Engineering Fundamentals", Newnes, 2nd edition, 2000.

V. REFERENCE BOOKS:

1. Matthew N O Sadiku, S V Kulkarni, "Principles of Electromagnetics", Oxford University Press, 6th edition, 2015.
2. AS Mahajan, AA Rangwala "Electricity And Magnetism", McGraw Hill Publications, 1st edition, 2000.
3. MS Naidu, V Kamaraju "High Voltage Engineering", McGraw Hill Publications, 3rd edition, 2013.
4. William H Hayt, John A Buck, "Problems and Solutions in Electromagnetics", McGraw Hill Publications, 1st edition, 2010.

VI. ELECTRONIC RESOURCES:

1. https://www.calvin.edu/~pribeiro/courses/engr315/EMFT_Book.pdf
2. <https://www.web.mit.edu/viz/EM/visualizations/coursenotes/modules/guide02.pdf>
3. <https://www.nptel.ac.in/courses/108106073/>
4. <https://www.iare.ac.in>

VIII. MATERIALS ONLINE:

1. Course template
2. Tutorial question bank
3. Tech talk topics
4. Open end experiments
5. Definitions and terminology
6. Assignments
7. Model question paper - I
8. Model question paper - II
9. Lecture notes
10. E-learning readiness videos (ELRV)
11. Power point presentation



INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal - 500 043, Hyderabad, Telangana

COURSE CONTENT

DATA STRUCTURES								
III Semester: AE / ME / CE / CSE / CSE (AI & ML) / CSE (DS) / CSE (CS) / IT / ECE / EEE								
Course Code	Category	Hours / Week			Credits	Maximum Marks		
ACSD08	Core	L	T	P	C	CIA	SEE	Total
		3	-	-	3	40	60	100
Contact Classes: 48	Tutorial Classes: Nil	Practical Classes: Nil			Total Classes: 48			
Prerequisite: Essentials of Problem Solving								

I. COURSE OVERVIEW:

The course covers some of the general-purpose data structures and algorithms, and software development. Topics covered include managing complexity, analysis, static data structures, dynamic data structures and hashing mechanisms. The main objective of the course is to teach the students how to select and design data structures and algorithms that are appropriate for problems that they might encounter in real life. This course reaches to student by power point presentations, lecture notes, and lab which involve the problem solving in mathematical and engineering areas.

II. COURSES OBJECTIVES:

The students will try to learn

- The skills needed to understand and analyze performance trade-offs of different algorithms / implementations and asymptotic analysis of their running time and memory usage.
- The basic abstract data types (ADT) and associated algorithms: stacks, queues, lists, tree, graphs, hashing and sorting, selection and searching.
- The fundamentals of how to store, retrieve, and process data efficiently.
- The implementing these data structures and algorithms in Python.
- The essential for future programming and software engineering courses.

III. COURSE OUTCOMES:

At the end of the course students should be able to:

- CO 1 Interpret the complexity of the algorithm using the asymptotic notations.
- CO 2 Select the appropriate searching and sorting technique for a given problem.
- CO 3 Construct programs on performing operations on linear and nonlinear data structures for organization of a data.
- CO 4 Make use of linear data structures and nonlinear data structures solving real-time applications.
- CO 5 Describe hashing techniques and collision resolution methods for accessing data with respect to performance.
- CO 6 Compare various types of data structures; in terms of implementation, operations and performance.

IV. COURSE CONTENT:

MODULE – I: INTRODUCTION TO DATA STRUCTURES, SEARCHING AND SORTING (09)

Basic concepts: Introduction to data structures, classification of data structures, operations on data structures, Algorithm Specification, Recursive algorithms, Data Abstraction, Performance analysis- time complexity and space complexity, Introduction to Linear and Non Linear data structures, Searching techniques: Linear and Binary search, Uniform Binary Search, Interpolation Search, Fibonacci Search; Sorting techniques: Bubble, Selection, Insertion, and Quick, Merge, Radix and Shell Sort and comparison of sorting algorithms.

MODULE – II: LINEAR DATA STRUCTURES (09)

Stacks: Stack ADT, definition and operations, Implementations of stacks using array, applications of stacks, Arithmetic expression conversion and evaluation; Queues: Primitive operations; Implementation of queues using

Arrays, applications of linear queue, circular queue and double ended queue (deque).

MODULE – III: LINKED LISTS (09)

Linked lists: Introduction, singly linked list, representation of a linked list in memory, operations on a single linked list; Applications of linked lists: Polynomial representation and sparse matrix manipulation.

Types of linked lists: Circular linked lists, doubly linked lists; Linked list representation and operations of Stack, linked list representation and operations of queue.

MODULE – IV: NON LINEAR DATA STRUCTURES (09)

Trees: Basic concept, binary tree, binary tree representation, array and linked representations, binary tree traversal, binary tree variants, threaded binary trees, application of trees, Graphs: Basic concept, graph terminology, Graph Representations.

Adjacency matrix, Adjacency lists, graph implementation, Graph traversals – BFS, DFS, Application of graphs, Minimum spanning trees – Prims and Kruskal algorithms.

MODULE – V: BINARY TREES AND HASHING (09)

Binary search trees: Binary search trees, properties and operations; Balanced search trees: AVL trees; Introduction to M- Way search trees, B trees; Hashing and collision: Introduction, hash tables, hash functions, collisions, applications of hashing.

V. TEXT BOOKS:

1. Rance D. Necaise, “Data Structures and Algorithms using Python”, Wiley Student Edition.
2. Benjamin Baka, David Julian, “Python Data Structures and Algorithms”, Packt Publishers, 2017.

VI. REFERENCE BOOKS:

1. S. Lipschutz, “Data Structures”, Tata McGraw Hill Education, 1st edition, 2008.
2. D. Samanta, “Classic Data Structures”, PHI Learning, 2nd edition, 2004.

VII. ELECTRONIC RESOURCES:

1. https://www.tutorialspoint.com/data_structures_algorithms/algorithms_basics.htm
2. <https://www.codechef.com/certification/data-structures-and-algorithms/prepare>
3. <https://www.cs.auckland.ac.nz/software/AlgAnim/dsToC.html>
4. <https://online-learning.harvard.edu/course/data-structures-and-algorithms>

VIII. MATERIALS ONLINE

1. Course template
2. Tutorial question bank
3. Definition and terminology
4. Tech-talk topics
5. Assignments
6. Model question paper - I
7. Model question paper - II
8. Lecture notes
9. Early learning readiness videos (ELRV)
10. Power point presentations



INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal - 500 043, Hyderabad, Telangana

COURSE CONTENT

ANALOG CIRCUITS								
III Semester: EEE								
Course Code	Category	Hours / Week			Credits	Maximum Marks		
AECD05	Core	L	T	P	C	CIA	SEE	Total
		3	0	0	3	40	60	100
Contact Classes: 48	Tutorial Classes: Nil	Practical Classes: Nil				Total Classes: 48		
Prerequisite: : Electrical Circuits , Linear Algebra and Calculus								

I. COURSE OVERVIEW:

This course provides the knowledge over the principles and construction of analog electronics. It covers the characteristics of electronic devices such as diodes, transistors, operational amplifiers and analysing amplifier circuits using small signal model and hybrid pi model, linear and nonlinear wave shaping. It focuses on applications in the area of power electronics, digital electronics and VLSI design.

II. COURSE OBJECTIVES:

The students will try to learn:

- The operational principles of analog electronic circuits such as feedback amplifiers and operational amplifiers
- The analog circuits fundamental theory to build signal conversion circuits, filter circuits, Data converters and Automatic Gain Control.
- The analog circuits applications in the advanced fields power electronics such as power factor monitoring
- circuits, power quality measurement, SMPS and battery controls.

III. COURSE OUTCOMES:

After successful completion of the course, students should be able to:

CO1	Recall the principles and operation of pn diode for the applications such as rectifiers, clippers, and clampers.
CO2	Illustrate the characteristics of bipolar and uni polar transistor for operating in different regions of operation.
CO3	Estimate feedback amplifiers parameters based on sampling and mixer circuits.
CO4	Determine frequency of oscillations for the RC, LC, Hartley and Colpitts oscillators.
CO5	Understand the principles and operations of operational amplifier for operating in different regions of operation.
CO6	Utilize inverting and non inverting amplifiers as waveform generators and in IC related real time applications.

IV. COURSE CONTENT:

MODULE - I: SEMICONDUCTOR DEVICES (10)

P-N junction diode, V-I characteristics of a diode, BJT construction, Input output characteristics Common Base, Common Emitter, Common Collector configurations, FET, MOSFET construction, drain and transfer characteristics.

MODULE –II: SEMICONDUCTOR DEVICE APPLICATIONS (09)

P-N diode applications: half-wave and full-wave rectifiers, clippers and clampers, BJT Load line analysis, common emitter, common base and common collector amplifiers; Small signal equivalent circuits. small signal model of FET, gain, input and output impedances, common-source, common-gate and common-drain amplifiers.

MODULE –III: FEEDBACK AMPLIFIERS AND OSCILLATORS (10)

Concepts of feedback: Classification of feedback amplifiers, general characteristics of negative feedback amplifiers, effect of feedback on amplifier characteristics, voltage series, voltage shunt, current series and current shunt feedback configurations, simple problems.

Oscillators: Condition for Oscillations, RC type Oscillators: RC phase shift and Wien-bridge Oscillators, LC type

Oscillators: generalized analysis of LC Oscillators, Hartley and Colpitts oscillators

MODULE – IV: OPERATIONAL AMPLIFIERS (10)

Operational Amplifier: Differential Amplifier, DC and AC analysis of dual input balanced output configuration, dual input unbalanced output. Characteristics of Op-amps, Op-amp block diagram, ideal and practical Op-amp specifications. DC characteristics: Input, output offset voltages and currents, drift. AC characteristics: Frequency response, slew rate, CMRR.

MODULE – V: OPERATIONAL AMPLIFIER APPLICATIONS (09)

Linear applications of Op-amps: Inverting and non-inverting amplifier, integrator, differentiator, instrumentation amplifier. Non-linear applications of Op-Amps: Comparators, monostable and astable multi vibrators, triangular, saw tooth, square wave generators, log and anti-log amplifiers.

V. TEXT BOOKS:

1. Jacob Millman, Christos C Halkias, “Integrated Electronics”, McGraw Hill Education, 2nd Edition 2010.
2. Ramakanth A, Gayakwad, “Op-Amps & Linear Ics”, PHI, 2003.

VI. REFERENCE BOOKS:

1. Thomas L. Floyd, “Electronic Devices Conventional and Current Version”, Pearson, 2013.
2. J. Millman and A. Grabel, “Microelectronics”, McGraw Hill Education, 1988.
3. P. Horowitz and W. Hill, “The Art of Electronics”, Cambridge University Press, 1989.
4. P. R. Gray, R. G. Meyer and S. Lewis, “Analysis and Design of Analog Integrated Circuits”, John Wiley & Sons, 2001

VII. ELECTRONIC RESOURCES:

1. <http://www-mdp.eng.cam.ac.uk/web/library/enginfo/electrical/hong1.pdf>
2. <https://archive.org/details/ElectronicDevicesCircuits>
3. http://nptel.ac.in/courses/Webcourse-contents/IIT-ROORKEE/BASIC ELECTRONICS/home_page.htm
4. www.nptel.ac.in
5. notes.specworld.in/pdc-pulse-and-digital-circuits

VIII. MATERIALS ONLINE:

1. Course template
2. Tutorial question bank
3. Tech talk topics
4. Open end experiments
5. Definitions and terminology
6. Assignments
7. Model question paper - I
8. Model question paper - II
9. Lecture notes
10. E-learning readiness videos (ELRV)
11. Power point presentation



INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal - 500 043, Hyderabad, Telangana

COURSE CONTENT

ELECTRICAL NETWORKS AND SIMULATION LABORATORY								
III Semester: EEE								
Course Code	Category	Hours / Week			Credits	Maximum Marks		
AEED08	Core	L	T	P	C	CIA	SEE	Total
		-	-	2	1	40	60	100
Contact Classes: Nil	Tutorial Classes: Nil	Practical Classes: 45			Total Classes: 45			
Prerequisite: Electrical Circuits, Linear Algebra and Calculus								

I. COURSE OVERVIEW:

The Network Analysis and Scientific Computing Laboratory is designed to give hands-on experience on virtual instrumentation through digital simulation techniques. These techniques enable the students in examining characteristics of DC and AC circuits, filters, solution of differential equation, generation of three phase and complex wave forms using MATLAB.

II. COURSES OBJECTIVES:

The students will try to learn

- Time varying characteristics of series and parallel circuits using MATLAB.
- Transfer function of electrical circuits using MATLAB.
- Relations between electrical quantities in complex electrical networks using MATLAB.
- The performance of single phase and three phase circuits using Lab View.

III. COURSE OUTCOMES:

At the end of the course students should be able to:

- CO 1 Identify the symbols, tool kits and connections in Simulink environment for computing the quantities associated with electrical circuits
- CO 2 Examine the transfer function for studying transient response of RL, RC and RLC circuits.
- CO 3 Analyze the virtual instrumentation (VI) using control loops, arrays, charts and graphs
- CO 4 Determine various alternating quantities of single phase and three phase signals generated in MATLAB/LabVIEW.
- CO 5 Exam Design the various sensors for measuring electrical and non-electrical quantities through digital simulation.

IV. COURSE CONTENT:

EXERCISES ON ELECTRICAL NETWORKS AND SIMULATION LABORATORY

Note: Students are encouraged to bring their own laptops for laboratory practice sessions.

1. Getting Started Exercises

Introduction to MATLAB

Identify the symbols, tool kits and connections in Simulink environment for computing the quantities associated with electrical circuits.

MATLAB is a powerful software environment widely used in various fields, including electrical engineering and circuit analysis. It provides a range of symbols, toolkits, and connections related to electrical circuits to assist engineers and researchers in designing, analyzing, and simulating electrical circuits.

Here's an introduction to some of the key features and tools MATLAB offers in this regard:

1. **Symbolic Math Toolbox:** MATLAB's Symbolic Math Toolbox allows you to perform symbolic computations, which are crucial for electrical circuit analysis. You can define symbolic variables for circuit components like resistors, capacitors, and inductors and manipulate algebraic equations symbolically. This is useful for solving circuit equations and obtaining transfer functions.
2. **Simulink/MATLAB:** Simulink is a graphical environment within MATLAB for modeling, simulating, and analyzing dynamic systems, including electrical circuits. It provides a vast library of pre-built electrical components and blocks, making it easier to create circuit models and simulate their behavior.
3. **Control System Toolbox:** This toolbox contains tools and functions for analyzing and designing control systems, which are essential for understanding the behavior of electrical circuits. You can use it to design controllers, analyze stability, and simulate the response of electrical systems.
4. **Electrical Circuit Analysis Toolbox:** There are also third-party toolboxes available for MATLAB, such as the Electrical Circuit Analysis Toolbox, that provide specialized tools for electrical circuit analysis. These toolboxes often include features like circuit simulation, AC/DC analysis, and more.
5. **Sim Power Systems (formerly known as Sim Electronics):** Sim Power Systems is a specialized toolbox in Simulink for modeling and simulating electrical circuits and systems. It includes a wide range of electrical components, such as transformers, generators, and motors, allowing you to create detailed electrical system models.
6. **Sim scape Electrical:** Sim scape Electrical is another Simulink toolbox that focuses on modeling and simulating electrical power systems. It includes components like power electronics devices, electrical machines, and power generation and distribution elements.
7. **Signal Processing Toolbox:** This toolbox is valuable for analyzing signals within electrical circuits. You can use it for tasks like filtering noisy signals, performing Fourier analysis, and extracting important information from sensor data.

8. **Data Acquisition Toolbox:** If you're working with hardware experiments or data acquisition systems, this toolbox can help you connect MATLAB to external hardware, making it easier to acquire and analyze data from real-world electrical circuits.
9. **Instrument Control Toolbox:** This toolbox enables you to communicate with and control external instruments like oscilloscopes, signal generators, and multimeters, which is beneficial for experimental work in electrical engineering.
10. **Custom Functions and Scripts:** MATLAB allows you to create custom functions and scripts to model and analyze specific aspects of electrical circuits. You can define your own functions for circuit equations, transfer functions, and more.

Creating syntax programs for electrical circuit analysis in MATLAB often involves using built-in functions, toolboxes, and custom code to define circuits, simulate their behavior, and analyze results. Below are some examples of syntax programs for common tasks in electrical circuit analysis using MATLAB.

Sample Exercises:

Hints

1. Defining Circuit Components and Equations:

```
% Define symbolic
variables syms V1 V2 R1 R2
C1 L1 s

Define resistor values
R1 = 100; % Ohms
R2 = 200; % Ohms

Define capacitor and inductor
values C1 = 0.1; % Farads
L1 = 0.5; % Henrys

Define Kirchhoff's laws for an RC
circuitEq1 = V1 - R1 * I1 - V2 == 0;
circuitEq2 = V2 - R2 * I2 - I1/C1 == 0;

Define Laplace domain equations
```

2. Solving Circuit Equations:

```
Solve the Laplace domain equations
symbolically I1_Laplace = solve (Eq1_Laplace,
I1);
I2_Laplace = solve (Eq2_Laplace, I2);

Inverse Laplace transform to get time-domain solutions
I1_time = ilaplace (I1_Laplace, s, t);
```

3. Simulating Circuits in Simulink:

```
% Create a Simulink model for an RLC circuit
model = 'RLC_Circuit';
open_system(new_system(model));

% Add components and connections using Simulink blocks
add_block ('Simulink/Continuous/Resistor', [model '/R1']);
add_block ('Simulink/Continuous/Inductor', [model '/L1']);
add_block ('Simulink/Continuous/Capacitor', [model '/C1']);
add_block ('Simulink/Sources/Step', [model '/Step Voltage']);
add_line (model, 'Step Voltage/1', 'R1/1');
% Add connections as needed

% Set component values and simulation parameters
set_param ([model '/R1'], 'Resistance', '100');
set_param ([model '/L1'], 'Inductance', '0.5');
set_param ([model '/C1'], 'Capacitance', '0.1');
set_param (model, 'Stop Time', '5');

% Run the simulation
sim(model);
```

4. Analyzing Frequency Response:

```
Define a transfer function for an RC circuit
num = [1];
den = [R1*C1 1]; sys =
tf (num, den);

Bode plot for frequency response
bode(sys);
```

5. Data Analysis and Plotting:

```
Import experimental data from a file
data = import data('experimental_data.csv');
time = data (: 1);
voltage = data (: 2);

Plot voltage-time graph
plot (time, voltage);
xlabel ('Time (s)');
ylabel ('Voltage (V)'); title
```

Try:

a. Circuit Equations:

- Write down the Kirchhoff's Voltage Law (KVL) equation for a series RC circuit involving a resistor (R) and a capacitor (C). Assume a constant voltage source V_s .
- Write down the Kirchhoff's Voltage Law (KVL) equation for a series RL circuit involving a resistor (R) and an inductor (L). Assume a time-varying current source $I_s(t)$.

b. MATLAB Code: Write MATLAB code to:

- Solve the KVL equation for the RC circuit (from 1.a (i)) symbolically for V_c as a function of time, assuming initial conditions.
- Create a Simulink model for the RL circuit (from 1.a (ii)) with appropriate components and

transient analysis for specified time duration.

Hint: Save your MATLAB code as a separate script file (.m) for questions

c. Transfer Function:

- i. Define the transfer function $H(s)$ for the RC circuit in terms of Laplace variables.
- ii. Plot the Bode plot of $H(s)$ to analyze the frequency response of the RC circuit.

Set

the

simulation

parameters

and

run

a

table

d. Data Analysis:

- i. Import experimental voltage and time data from a CSV file into MATLAB from (from 1.b (i), 1.b(ii))
- ii. Plot the voltage-time graph and label the axes appropriately.

e. Circuit Toolbox:

Mention at least two MATLAB toolboxes that are useful for electrical circuit analysis. Explain briefly how each toolbox can assist in circuit analysis.

- f.** If $R = 10\ \Omega$ and the current is increased from 0 to 10 A with increments of 2A, write a MATLAB program to generate a table of current, voltage and power dissipation.

2. Transient Response of Series RL, RC and RLC Circuits:

Examine the time varying characteristics of series RL, RC and RLC circuits for given values of R, L and C using MATLAB software.

2.1 Transient Response of Series RL

In this MATLAB program:

- i. Circuit parameters (resistance, inductance, and applied voltage) are defined.
- ii. The time constant (τ) is calculated as L / R .
- iii. A symbolic variable t is defined for time.
- iv. The differential equation representing the transient response of the RL circuit is defined using **diff** and **==**.
- v. The **dsolve** function is used to solve the differential equation symbolically.
- vi. A time vector (**'t_span'**) is created for the simulation.
- vii. $IL(t)$ is numerically evaluated for the given time vector.
- viii. The transient response is plotted.

Hints

```
% Define circuit parameters
R = 100; % Resistance in ohms
L = 0.5; % Inductance in henrys
Vin = 10; % Applied voltage in volts

% Calculate the time constant  $\tau$ 
tau = L / R;

% Define the symbolic variable for time
syms t

% Define the differential equation for the current  $IL(t)$ 
IL = sym('IL(t)');
eqn = L * diff(IL, t) + R * IL == Vin;

% Solve the differential equation
IL_solution = dsolve (eqn, IL);
IL_solution = simplify(IL_solution);

% Create a time vector and evaluate  $IL(t)$ 
time = 0:0.01:2*tau; % Adjust the time span as needed
IL_numeric = double (subs (IL_solution, t, time));

% Plot the transient response
plot (time, IL_numeric);
xlabel ('Time (s)');
ylabel ('Current (A)');
```

```
title ('Transient Response of RL Circuit');  
grid on;
```

Try:

a. Circuit Parameters:

Given the following values:

- i. Resistor (R): 150 ohms
- ii. Inductor (L): 0.3 H
- iii. Applied Voltage (V_{in}): 12 V

Calculate the time constant (τ) of the RL circuit.

b. Transient Analysis:

- iv. Write down the first-order differential equation governing the transient response of the RL circuit in terms of the current (I_L) through the inductor.
- v. Solve the differential equation symbolically to obtain the expression for $I_L(t)$ as a function of time (t).

c. MATLAB Simulation: Write MATLAB code to:

- i. Define the circuit parameters (R, L, and V_{in}).
- ii. Simulate the transient response of the RL circuit using the ' I_{sim} ' function. Use a time vector from 0 to 5τ with a suitable time step.
- iii. Plot the transient response of the current (I_L) as a function of time. Label the axes appropriately.

d. Time Constant Verification:

- i. Calculate the time constant τ numerically from the transient response obtained in question 2.1(c).
- ii. Compare the numerically calculated time constant with the value obtained in question 1. Explain any differences or similarities.

e. Analysis:

Based on the transient response plot obtained in question iii, discuss the behavior of the current in the RL circuit as it responds to the voltage step input. Specifically, explain the time constant, the initial current, and the behavior as time progresses.

2.2 Transient Response of Series RC

In this MATLAB code:

- i. Circuit parameters such as resistance (R), capacitance (C), and applied voltage (V_{in}) are defined.
- ii. The time constant (τ) is calculated as $R * C$.
- iii. A symbolic variable t is defined for time.
- iv. The differential equation representing the transient response of the RC circuit is defined using `diff` and `==`.
- v. The `dsolve` function is used to solve the differential equation symbolically.
- vi. A time vector (`t_span`) is created for the simulation.
- vii. $V_c(t)$ is numerically evaluated for the given time vector.
- viii. The transient response is plotted, showing how the voltage across the capacitor changes over time in response to a step input voltage.

You can adjust the time span in `t_span` to observe the transient response for the desired duration.

Hints

```
Define circuit parameters
R = 100; % Resistance in
ohms
C = 0.1; % Capacitance in farads
Vin = 5; % Applied voltage in
volts

Calculate the time constant  $\tau$ 
tau = R * C;

Define the symbolic variable for
timesyms t

Define the differential equation for the voltage VC(t) across the
capacitorVC = sym('VC(t)');
eqn = R * diff (VC, t) + VC == Vin;

Solve the differential equation
symbolicallyVC_solution = dsolve (eqn,
VC);
VC_solution = simplify(VC_solution);

Create a time vector
t_span = 0:0.01:5*tau; % Adjust the time span as needed

Evaluate VC(t) for the given time vector
VC_numeric = double (subs (VC_solution, t,
t_span));
```

Try:

a. Circuit Parameters:

Given the following values:

- i. Resistor (R): 220 ohms
- ii. Inductor (L): 0.01 F
- iii. Applied Voltage (Vin): 10 V

Calculate the time constant (τ) of the RC circuit.

b. Transient Analysis:

- i. Write down the first-order differential equation governing the transient response of the RC circuit in terms of the voltage (V_c) across the capacitor.
- ii. Solve the differential equation symbolically to obtain the expression for $V_c(t)$ as a function of time (t).

c. MATLAB Simulation: Write MATLAB code to:

- i. Define the circuit parameters (R, C, and Vin).
- ii. Simulate the transient response of the RC circuit using the 'l' function. Use a time vector from 0 to 5τ with a suitable time step.

- iii. Plot the transient response of the voltage across the capacitor (V_c) as a function of time. Label the axes appropriately.

d. Time Constant Verification:

- i. Calculate the time constant τ numerically from the transient response obtained in question 2.2c.
- ii. Compare the numerically calculated time constant with the value obtained in question 2.2(a). Explain any differences or similarities.

e. Analysis:

Based on the transient response plot obtained in question c(iii), discuss the behavior of the current in the RL circuit as it responds to the voltage step input. Specifically, explain the time constant, the initial current, and the behavior as time progresses.

2.3 Transient Response of Series RLC

In this MATLAB code:

- I. Circuit parameters such as resistance (R), inductance (L), capacitance (C), and applied voltage (V_{in}) are defined.
- II. The natural frequency (ω_n) and damping ratio (ζ) are calculated based on R , L , and C .
- III. The time constant (τ) is calculated as $1 / (\omega_n * \sqrt{1 - \zeta^2})$.
- IV. A symbolic variable ' t ' is defined for time.
- V. The differential equation representing the transient response of the RLC circuit is defined using '**diff**' and '='.
- VI. The **dsolve** function is used to solve the differential equation symbolically.
- VII. A time vector ('**t_span**') is created for the simulation.
- VIII. $V_C(t)$ is numerically evaluated for the given time vector.
- IX. The transient response is plotted, showing how the voltage across the capacitor changes over time in response to the applied voltage.

You can adjust the time span in t_span as needed to observe the transient response for the desired duration.

Hints

```
% Define circuit parameters
R = 100; % Resistance in ohms
L = 0.2; % Inductance in henrys
C = 0.01; % Capacitance in farads Vin
= 10; % Applied voltage in volts

% Calculate the natural frequency (wn) and damping ratio (zeta)wn =
1 / sqrt(L * C);
zeta = R / (2 * L * wn);

% Calculate the time constant (τ) tau
= 1 / (wn * (1 - zeta^2) ^0.5);

% Define the symbolic variable for time
syms t

% Define the differential equation for the voltage VC(t) across the capacitor
VC = sym('VC(t)');
eqn = L * diff (VC, t, 2) + R * diff (VC, t) + (1/C) * VC == Vin;

% Solve the differential equation symbolically
VC_solution = dsolve (eqn, VC);
VC_solution = simplify(VC_solution);

% Create a time vector
t_span = 0:0.001:5*tau; % Adjust the time span as needed

% Evaluate VC(t) for the given time vector VC_numeric
= double (subs (VC_solution, t, t_span));

% Plot the transient response
plot (t_span, VC_numeric);
xlabel ('Time (s)');
ylabel ('Voltage across Capacitor (V)'); title
('Transient Response of RLC Circuit');grid on;
```

Try

a. Circuit Parameters:

Given the following values:

- i. Resistor (R): 220 ohms
- ii. Inductor (L): 0.1 H
- iii. Capacitor (C): 0.01 F
- iv. Applied Voltage (Vin): 12 V

Calculate the natural frequency (ω_n), damping ratio (zeta), and time constant (τ) of the RLC circuit.

b. Transient Analysis:

- i. Write down the second-order differential equation governing the transient response of the RLC circuit in terms of the voltage (V_C) across the capacitor.
- iii. Solve the differential equation symbolically to obtain the expression for $V_C(t)$ as a function of time (t).

c. MATLAB Simulation: Write MATLAB code to:

- i. Define the circuit parameters (R, L, C, and V_{in}).
- ii. Simulate the transient response of the RLC circuit using the `lsim` function. Use a time vector from 0 to 5τ with a suitable time step.
- iii. Plot the transient response of the voltage across the capacitor (V_C) as a function of time. Label the axes appropriately.

d. Time Constant Verification:

- i. Calculate the natural frequency (ω_n), damping ratio (zeta), and time constant (τ) numerically from the transient response obtained in question 2.3(c).
- ii. Compare the numerically calculated values with the values obtained in question 2.3(a). Explain any differences or similarities.

3. Solving Differential Equations

Perform the solution of differential equation which is representing mathematical model of electric circuit using MATLAB software.

Define the Differential Equation: First, define the differential equation that represents the circuit. This equation depends on the specific circuit components and the mathematical model you're using. For instance, consider an RC circuit with a resistor (R) and a capacitor (C) connected in series. The differential equation for the voltage across the capacitor (V_C) can be written as:

$$RC \frac{dV_C}{dt} + V_C = V_{in}$$

where RC is the time constant, $\frac{dV_C}{dt}$

represents the derivative of V_C with respect to time, and V_{in} is

the applied voltage.

Convert to Symbolic Variables: In MATLAB, convert the equation into symbolic form using the '**syms**' function and define the symbolic variables. For example:

Hints

```
syms Vc(t) Vin R C
in = R * C * diff(Vc(t), t) + Vc(t) == Vin;
```

Solve the Differential Equation: Use the '**dsolve**' function to solve the differential equation symbolically:

```
Vc_solution = dsolve(eqn);
```

Define Initial Conditions: If your circuit has initial conditions (e.g., $V_c(0) = 0$), you can specify them in the

```
Vc_solution = dsolve(eqn, Vc(0) == 0); % Example initial condition
```

'**dsolve**' function to get a more specific solution.

Plot the Solution: Create a time vector and evaluate the symbolic solution for $V_c(t)$ for the given time vector. Then, plot the transient response:

```
t = 0:0.01:5*RC; % Adjust the time span as needed
Vc_numeric = double (subs (Vc_solution, t));
plot (t, Vc_numeric);
xlabel ('Time (s)');
ylabel ('Voltage across Capacitor (V)');
title ('Transient Response of RC Circuit');
```

Try

- Write down the differential equation representing the voltage across the capacitor (V_c) in the RC circuit. Include all the relevant terms and variables.
- Define the symbolic variables and the differential equation in MATLAB to represent the RC circuit.
- Use the **dsolve** function to solve the differential equation symbolically for $V_c(t)$. Assume an initial condition $V_c(0) = 0$.
- Write MATLAB code to:
Define the circuit parameters:
Resistance R: 100 ohms Capacitance C: 0.01 F Applied Voltage V_{in} : 10 V
- Simulate and plot the transient response of the voltage across the capacitor $V(t)$ for a time span of 0 to 5 times the time constant (τ).
- Label the axes appropriately and provide a title for the plot.
- Calculate the time constant (τ) of the RC circuit using the given values of R and C.
Based on the transient response plot obtained in question v, discuss the behavior of the voltage across the capacitor (V_c) as it responds to the voltage step input. Explain the concept of the time constant (τ) and its significance in the transient response.

3. Transfer Function of Electrical Circuit

Determine the voltage transfer function of series RLC electrical circuit for frequencies from 10Hz to 100kHz using MATLAB software.

To determine the voltage, transfer function of a series RLC electrical circuit for a range of frequencies from 10 Hz to 100 kHz using MATLAB, you can follow these steps

Define Circuit Parameters:

- Resistance (R) in ohms
- Inductance (L) in henrys
- Capacitance (C) in farads

Define Symbolic Variables:

- s as the complex frequency variable

Create the Transfer Function:

The transfer function (H(s)) for a series RLC circuit can be defined as follows:

$$H(s) = \frac{V_{out}(s)}{V_{in}(s)} = \frac{1}{LC^2 + RCs + 1}$$

Where:

$V_{ou}(s)$ is the Laplace-transformed output voltage?

$V_i(s)$ is the Laplace-transformed input voltage? L is the inductance.

C is the capacitance. R is the resistance.

s is the complex frequency variable. Calculate and Plot the Frequency Response:


```

% Define circuit
parametersR = 100; %
Resistance in ohms
L = 0.1; % Inductance in henrys
C = 0.01e-6; % Capacitance in farads (1 uF)
frequencies = logspace (1, 5, 100); % Frequency range from 10 Hz to 100 kHz

% Define the complex frequency
variable ss = j*2*pi*frequencies;

% Calculate the transfer function
H(s)H = 1. / (L*C*s.^2 + R*C*s +
1);

% Calculate magnitude and phase of
H(s)magnitude_H = abs(H);
phase_H = angle(H);

% Plot the Bode plot (magnitude and
phase)subplot (2,1,1);
semilogx (frequencies, 20*log10(magnitude_H));
title ('Bode Plot of RLC
Circuit');xlabel ('Frequency
(Hz)');
ylabel ('Magnitude
(dB)');grid on;

subplot (2,1,2);

```

HINTS

In this code:

- We define the circuit parameters (R, L, and C).
- We create a range of frequencies from 10 Hz to 100 kHz using '**logspace**'.
- We define the complex frequency variable s as $j\omega$, where ω is the angular frequency.
- The transfer function ($H(s)$) is calculated for each frequency in the range.
- We calculate the magnitude and phase of $H(s)$
- Finally, we plot the Bode plot, showing the magnitude and phase response of the RLC circuit as a function of frequency.

Try

- i) A series RLC electrical circuit with the following parameters: Resistance (R): 150 ohms Inductance (L): 0.2 H Capacitance (C): 0.01 F
Determine the voltage transfer function ($H(s)$) of the circuit for a range of frequencies from 10 Hz to 100 kHz.
- ii) Write down the equation for the voltage transfer function ($H(s)$) of the series RLC circuit in terms of R, L, C and the complex frequency variable
- iii) Write MATLAB code to:
Define the given circuit parameters (R, L, and C).
Generate a range of frequencies from 10 Hz to 100 kHz.
Calculate the complex frequency variable s for each frequency in the range.
Calculate the voltage transfer function ($H(s)$) for each frequency using the equation from question II.
- iv) Magnitude and Phase Plot: Write MATLAB code to:
Calculate the magnitude and phase of $H(s)$ for the entire frequency range.
Plot the Bode plot of the magnitude (in decibels) and phase (in degrees) of $H(s)$ as a function of frequency.
Based on the Bode plot obtained in question IV(b), analyze and discuss the behavior of the series RLC circuit as the frequency changes from 10 Hz to 100 kHz. Explain how the circuit responds to different frequencies and highlight any resonance frequencies if observed.

3. Transient Response of Parallel RL, RC and RLC Circuits

Examine the time varying characteristics of parallel RL, RC and RLC circuits for given values of R, L and C using MATLAB software.

3.1 Transient Response of Parallel RL

1. Define Circuit Parameters:

- Resistance (R) in ohms
- Inductance (L) in henrys
- Applied Voltage (V_{in}) in volts

2. Define Symbolic Variables:

- t as the time variable
- $I_L(t)$ as the current through the inductor as a function of time

3. Create the Differential Equation:

The differential equation for a parallel RL circuit can be written as:

$$L \frac{dI(t)}{dt} + V_m = 0$$

Where:

- L is the inductance.
- R is the resistance.
- $I(t)$ is the current through the inductor as a function of time.
- V_{in} is the applied voltage.

$dt \ R$

4. Solve the Differential Equation:

Use the '**dsolve**' function to solve the differential equation symbolically:

Hints

```
Define the differential equation
syms t
IL = sym('IL(t)');
eqn = L * diff (IL, t) + Vin / R == 0;

Solve the differential equation symbolically
IL_solution = dsolve (eqn, IL);
IL_solution = simplify(IL_solution);
```

Create a Time Vector and Evaluate the Solution:

```
Create a time vector
t_span = 0:0.01:5*tau; % Adjust the time span as needed

Evaluate IL(t) for the given time vector
IL_numeric = double (subs (IL_solution, t, t_span));

Create a time vector and evaluate I(t) for the given time vector:
```

Plot the Transient Response:

Plot the transient response of the current $I(t)$ as a function of time:

```
Plot the transient response
plot (t_span, IL_numeric);
xlabel ('Time (s)');
ylabel ('Current (A)');
title ('Transient Response of Parallel RL Circuit'); grid on;
```

In this code:

1. Define the circuit parameters (R , L , and V_{in}).
2. Create a symbolic variable t for time and a symbolic variable $I(t)$ for the current through the inductor.
3. Define the differential equation and solve it symbolically.
4. Create a time vector for the simulation.
5. Evaluate the symbolic solution for $I(t)$ numerically.
6. Finally, we plot the transient response of the current.

Try

1. Circuit Parameters:

Given the following values:

Resistor (R): 220
ohms Inductor
(L): 0.1 H

Applied Voltage (V_{in}): 12 V

Calculate the time constant (τ) of the parallel RL circuit.

2. Transient Analysis:

- a) Write down the first-order differential equation governing the transient response of the RL circuit in terms of the current $I(t)$ through the inductor.
- b) Solve the differential equation symbolically to obtain the expression for $I(t)$ as a function of time (t).

3. MATLAB Simulation:

1. Write MATLAB code to:

- Define the circuit parameters (R , L , and V_{in}).
- Simulate the transient response of the RL circuit using the '**lsim**' function. Use a timevector from 0 to 5τ with a suitable time step.
- Plot the transient response of the current I_L through the inductor as a function of time. Label the axes appropriately.

2. Time Constant Verification:

- Calculate the time constant (τ) numerically from the transient response obtained in question.
- Compare the numerically calculated time constant with the value obtained in question
- Explain any differences or similarities.

3. Analysis:

Based on the transient response plot obtained in question, discuss the behavior of the current (I_L) in the RL circuit as it responds to the voltage step input. Specifically, explain the time constant, the initial current, and the behavior as time progresses.

4. Transient Response of Parallel RC

To analyze the transient response of a parallel RC circuit in MATLAB, you can use the circuit's differential equation and solve it numerically. The transient response describes how the circuit behaves as it moves from one steady state to another when an input voltage or current changes suddenly.

Assuming you have a parallel RC circuit with a voltage source connected in parallel to a resistor (R) and a capacitor (C), you can write the differential equation for this circuit as follows:

$$V(t) = I(t) \cdot R + V_C(t)$$

Where:

$V(t)$ is the voltage across the circuit as a function of time. $I(t)$ is the current through the resistor as a function of time.

$V_C(t)$ is the voltage across the capacitor as a function of time.

You can use MATLAB to solve this differential equation and obtain the transient response.

Hints

1. Define the circuit parameters (R and C) and the input voltage waveform (V_in) as functions of time.

```
R = 1; % Resistor value in ohms
C = 0.1; % Capacitor value in farads
```

```
Define the input voltage as a function of time (for example, a step
input)t = 0:0.01:5; % Time vector
V_in = zeros(size(t)); % Initialize the input voltage
V_in(t >= 0) = 1; % Set the input voltage to 1V
for t >= 0
```

```
% Define circuit parameters
R = 220; % Resistance in ohms
C = 0.01 % Capacitance in farads
V_in = 12;% Applied voltage in volts
```

```
% Calculate the time constant (tau)
tau = R * C;
```

```
% Define the symbolic variable for time
syms t
```

```
% Define the differential equation for the voltage VC(t) across the capacitor
VC = sym('VC(t)');
eqn = C * diff(VC, t) + VC / R == V_in;
```

```
% Solve the differential equation symbolically
VC_solution = dsolve(eqn, VC);
VC_solution = simplify(VC_solution);
```

```
% Create a time vector
```

```
Define the differential equation for the parallel RC circuit
dVc/dt = (V_in - Vc) / (R * C)
Where Vc is the voltage across the capacitor as a function of
time
dVc_dt = @(t, Vc) (V_in(t) - Vc) / (R * C);
```

```
Initial condition (at t = 0, Vc =
Vc0 = 0;
```

```
Solve the differential equation using
ode45[t, Vc] = ode45(dVc_dt, t, Vc0);
```

```
Plot the transient response
figure;
plot(t, Vc);
xlabel('Time
(s)');
ylabel('Voltage across Capacitor (V)');
title('Transient Response of Parallel RC
Circuit');grid on;
```

1. Plot the transient response of the voltage across the capacitor.
2. To simulate and analyze the transient response of a parallel RC (Resistor-Capacitor) circuit in MATLAB, you can use the following syntax:

```
t_span = 0:0.001:5*tau; % Adjust the time span as needed

% Evaluate VC(t) for the given time vector
VC_numeric = double (subs (VC_solution, t, t_span));

% Plot the transient response
plot (t_span, VC_numeric);
xlabel ('Time (s)');
ylabel ('Voltage across Capacitor (V)');
title ('Transient Response of Parallel RC Circuit');
grid on;
```

Try:

1. Circuit Parameters:

1. Given the following values:
2. Resistor (R): 100 ohms
Capacitance (C): 0.01 F
Applied Voltage (V_{in}): 10 V
3. Calculate the time constant (τ) of the parallel RC circuit.

2. Transient Analysis:

- a) Write down the first-order differential equation governing the transient response of the RC circuit in terms of the voltage (V_c) across the capacitor.
- b) Solve the differential equation symbolically to obtain the expression for $V_c(t)$ as a function of time (t).

3. MATLAB Simulation:

Write MATLAB code to:

- a) Define the circuit parameters (R, C, and V_{in}).
- b) Simulate the transient response of the RC circuit using the '**lsim**' function. Use a time vector from 0 to 5τ with a suitable time step.
- c) Plot the transient response of the voltage (V_c) across the capacitor as a function of time. Label the axes appropriately.

4. Time Constant Verification:

- a) Calculate the time constant (τ) numerically from the transient response obtained in question.
- b) Compare the numerically calculated time constant with the value obtained in question.
- c) Explain any differences or similarities.

5. Transient Response of Parallel RLC

To calculate the transient response of a parallel RLC circuit in MATLAB, you can use the circuit's differential equation and solve it numerically. Here's an example of how you can do this: Assuming you have a parallel RLC circuit with the following parameters:

Resistance (R) Inductance (L) Capacitance (C) Voltage source (V)

Initial conditions (initial voltage across the capacitor and initial current through the inductor)

Here's the MATLAB code to calculate and plot the transient response of the parallel RLC circuit:

Hints

```
% Define circuit parameters
R = 1;          % Resistance (in ohms)
L = 0.5;        % Inductance (in henries)
C = 0.2;        % Capacitance (in farads)
V = 5;          % Voltage source (in volts)

% Define initial conditions
Vc0 = 0;        % Initial voltage across the capacitor (in volts)
I10 = 0;        % Initial current through the inductor (in amperes)

% Create a time vector
t = 0:0.01:5;   % Time from 0 to 5 seconds with a step of 0.01 seconds

% Define the differential equation for the parallel RLC circuit
% d2Vc/dt^2 + (1/(R*C)) * dVc/dt + (1/(L*C)) * Vc = (1/L) * dI1/dt
% d2I1/dt^2 + (1/(R*L)) * dI1/dt + (1/(L*C)) * I1 = (1/L) * dVc/dt
% Where Vc is the voltage across the capacitor and I1 is the current through the inductor

% Define a function that returns the derivatives of Vc and I1
dVcDt = @(t, Vc, I1) (1/L) * (V - Vc) - (1/(R*C)) * Vc;
dI1Dt = @(t, Vc, I1) (1/L) * (Vc - V) - (1/(R*L)) * I1;

% Use the ODE solver to solve the system of differential equations
initial_conditions = [Vc0, I10];
[t, y] = ode45(@(t, y) [dVcDt(t, y(1), y(2)); dI1Dt(t, y(1), y(2))], t, initial_conditions);

% Extract voltage across the capacitor and current through the inductor
Vc = y(:, 1);
I1 = y(:, 2);

% Plot the transient response
subplot(2, 1, 1);
plot(t, Vc);
xlabel('Time (s)');
ylabel('Voltage across Capacitor (V)');
title('Transient Response of Parallel RLC Circuit'); grid on;
```


Try:

1. Mathematical Modeling:

- a) Consider a parallel RLC circuit with the following parameters:
 - Resistance (R): 2 ohms
 - Inductance (L): 0.5 henries
 - Capacitance (C): 0.1 farads
 - Voltage source (V): 10 volts
 - b) Define the initial conditions as follows:
 - Initial voltage across the capacitor ($V_C(0)$): 0 volts
 - Initial current through the inductor ($I_L(0)$): 0 amperes
 - c) Write the differential equations that describe the behavior of this parallel RLC circuit over time. Express these equations in terms of voltage across the capacitor (V_C) and current through the inductor (I_L).
2. Using MATLAB, simulate and plot the transient response of the parallel RLC circuit for the following conditions:
- Time span: 0 to 5 seconds
 - Time step: 0.01 seconds

Create MATLAB scripts/functions to do the following:

- a) Define the circuit parameters and initial conditions as specified in Problem 1.
 - b) Create a time vector that covers the time span.
 - c) Implement the differential equations for the parallel RLC circuit.
 - d) Use MATLAB's ODE solver (ode45) to solve the differential equations.
 - e) Plot the transient response, showing both the voltage across the capacitor and the current through the inductor on separate graphs.
3. Analysis and Interpretation
- a) Analyze the plots obtained in Problem 2. Explain how the voltage across the capacitor and the current through the inductor change over time during the transient response.
 - b) Calculate and report the time constant (τ) of this parallel RLC circuit based on its parameters (R, L, and C). Discuss the significance of the time constant in relation to the transient response.

6. Generation of Three Phase Wave Form

In this code:

- Parameters such as frequency and amplitude are defined.
- Phase differences and phase sequences are specified in degrees.
- A time vector (t) is created to represent time.
- Phase angles in radians are calculated for each phase based on time and phase differences.
- Sinusoidal waveforms for each phase (Phase A, Phase B, and Phase C) are generated.
- Finally, the waveforms are plotted on separate subplots to visualize the three-phase AC waveform.
- You can adjust the 'frequency', 'amplitude', 'phase_diff_AB', 'phase_diff_BC', and 'phase_diff_CA' values to observe different phase differences and phase sequences in the generated waveform.

Hints

```
% Define parameters
frequency = 50; % Frequency in Hertz
amplitude = 220; % Amplitude of each phase in volts

% Define phase differences and phase sequences
phase_diff_AB = 0; % Phase difference between phases A and B (in degrees)
phase_diff_BC = 120; % Phase difference between phases B and C (in degrees)
phase_diff_CA = 240; % Phase difference between phases C and A (in degrees)

% Create time vector
t = 0:0.0001:0.1; % Time vector (0 to 0.1 seconds with a step of 0.0001 seconds)

% Calculate phase angles in radians
omega = 2 * pi * frequency;
theta_A = omega * t;
theta_B = omega * t - deg2rad(phase_diff_AB);
theta_C = omega * t - deg2rad(phase_diff_CA);

% Generate sinusoidal waveforms for each phase
phase_A = amplitude * sin(theta_A);
phase_B = amplitude * sin(theta_B);
phase_C = amplitude * sin(theta_C);

% Plot the three-phase waveforms
figure;
subplot(3, 1, 1);
plot(t, phase_A, 'b');
xlabel('Time (s)');
ylabel('Phase A Voltage (V)'); title('Three-Phase AC Waveform'); grid on;

subplot(3, 1, 2);
plot(t, phase_B, 'r');
xlabel('Time (s)');
ylabel('Phase B Voltage (V)'); grid on;

subplot(3, 1, 3);
plot(t, phase_C, 'g');
xlabel('Time (s)');
ylabel('Phase C Voltage (V)'); grid on;

% Adjust plot settings for better visualization
linkaxes;

% Legend for phase sequence
legend('Phase A', 'Phase B', 'Phase C');
```

Try:

Problem 1: Mathematical Background

Explain the concept of three-phase AC power systems, including the significance of phase differences and phase sequences. Discuss how the phase differences and sequences impact the operation of electrical systems.

Problem 2: MATLAB Programming

Write MATLAB code to generate three-phase AC waveforms with the following specifications:

- Frequency: 50 Hz
- Amplitude: 220 V

You should generate three phases (Phase A, Phase B, and Phase C) with different phase differences and sequences. Use a time vector with a duration of at least 0.1 seconds and a suitable time step.

- Implement code to create sinusoidal waveforms for each phase with the specified frequency and amplitude.
- Generate waveforms for the following scenarios:
 - Scenario 1: Balanced three-phase AC with a phase sequence of ABC and zero phase differences.
 - Scenario 2: Balanced three-phase AC with a phase sequence of BCA and phase differences of 120 degrees between phases.
 - Scenario 3: Unbalanced three-phase AC with a phase sequence of CBA and phase differences of 45 degrees between phases.
 - Plot each scenario's waveforms on separate graphs, clearly labeling the phases and including appropriate axis labels and titles.

Problem 3: Analysis and Discussion

- Analyze and compare the waveforms generated in Scenario 1, Scenario 2, and Scenario 3. Explain how the phase differences and phase sequences affect the shape and behavior of the three-phase AC waveforms.
- Discuss the practical applications of different phase sequences in electrical systems. Provide examples of where each phase sequence might be used and why.

Submission Instructions:

- 1) Provide well-commented MATLAB code for generating the three-phase AC waveforms (Problem 2).
- 2) Write a clear and concise analysis and discussion of the waveforms and phase sequences (Problem 3) in a separate document.
- 3) Submit both your MATLAB code and the analysis document as your assignment.

7. Measurement of three phase wave form parameters

In this code:

- Generate three-phase sinusoidal waveforms with the specified frequency and amplitude. Phase B and Phase C are 120 degrees out of phase with respect to Phase A, representing a balanced three-phase system.
- RMS voltage for each phase is calculated using the '**rms**' function.
- Phase angles between Phase A and other phases (Phase B and Phase C) are calculated using the '**atan2**' function and converted from radians to degrees.
- Total power is calculated for each phase assuming a known impedance, and then the total power is summed to get the total power in the three-phase system.
- You can adjust the '**frequency**', '**amplitude**', and '**impedance**' values to match your specific waveform and system parameters. This code will help you determine essential electrical quantities for a three-phase waveform in MATLAB.

Hints

```
% Define the three-phase waveform data
time = 0:0.0001:0.1; % Time vector (0 to 0.1 seconds with a step of 0.0001 seconds)
frequency = 50; % Frequency in Hertz
amplitude = 220; % Amplitude of each phase in volts

% Generate the three-phase sinusoidal waveforms
omega = 2 * pi * frequency;
phase_A = amplitude * sin(omega * time);
phase_B = amplitude * sin(omega * time - 2*pi/3); % Phase B is 120 degrees ahead of Phase A
phase_C = amplitude * sin(omega * time + 2*pi/3); % Phase C is 120 degrees behind Phase A

% Calculate RMS voltage for each phase
rms_A = rms(phase_A);
rms_B = rms(phase_B);
rms_C = rms(phase_C);

% Calculate phase angles between Phase A and other phases
phase_angle_B = rad2deg(atan2(imag(phase_B), real(phase_B)));
phase_angle_C = rad2deg(atan2(imag(phase_C), real(phase_C)));

% Calculate total power
power_A = rms_A^2 / impedance; % Assuming a known impedance
power_B = rms_B^2 / impedance;
power_C = rms_C^2 / impedance;
total_power = power_A + power_B + power_C;

% Display the results
fprintf('RMS Voltage (Phase A): %.2f V\n', rms_A);
fprintf('RMS Voltage (Phase B): %.2f V\n', rms_B);
fprintf('RMS Voltage (Phase C): %.2f V\n', rms_C);
fprintf('Phase Angle (Phase B - Phase A): %.2f degrees\n', phase_angle_B);
fprintf('Phase Angle (Phase C - Phase A): %.2f degrees\n', phase_angle_C);
fprintf('Total Power: %.2f Watts\n', total_power);
```

Try:

Problem 1: Mathematical Background

- I. Provide a brief explanation of three-phase electrical systems, including the significance of balanced and unbalanced systems. Discuss key electrical quantities such as RMS voltage, phase angles, and power in the context of three-phase waveforms.
- II. Modify the barrel shifter by changing the number of data lines to 16 bit wide to support 16 different functionalities

Problem 2: MATLAB Programming

Write MATLAB code to generate and analyze a three-phase waveform with the following specifications:

- Frequency: 60 Hz
 - Amplitude: 220 V (for each phase)
 - Duration: 0.2 seconds
- a. Generate three sinusoidal waveforms (Phase A, Phase B, and Phase C) with the specified frequency and amplitude. Phase B and Phase C should have phase differences of 120 degrees with respect to Phase A.
 - b. Calculate and display the following electrical quantities:
 - RMS voltage for each phase.
 - Phase angles between Phase A and Phase B, and between Phase A and Phase C.
 - Total power in the three-phase system. Assume a balanced system and a known impedance.

Problem 3: Analysis and Interpretation

- a. Analyze and discuss the results obtained in Problem 2. Explain the significance of RMS voltage, phase angles, and total power in a three-phase system.
- b. Consider a scenario where one of the phases (e.g., Phase B) becomes unbalanced and has a different amplitude (e.g., 200 V) compared to the other phases. Modify the MATLAB code to simulate this unbalanced scenario and recalculate the electrical quantities. Discuss how the unbalance affects the system.

Submission Instructions:

1. Provide well-commented MATLAB code for generating and analyzing the three-phase waveform (Problem 2).
2. Write a clear and concise analysis and interpretation of the electrical quantities and the impact of unbalance (Problem 3) in a separate document.
3. Submit both your MATLAB code and the analysis document as your assignment.

7. Virtual instruments (vi) using LabVIEW

Virtual instruments (VIs) in LabVIEW (Laboratory Virtual Instrument Engineering Workbench) are a powerful way to create customized software applications for data acquisition, analysis, and instrument control. VIs are graphical representations of an instrument or a measurement system and can be created using LabVIEW intuitive programming environment. Here are the steps to create a basic virtual instrument (VI) using LabVIEW.

% Launch LabVIEW

Start LabVIEW on your computer.

% Open a New VI

Click on "File" in the menu bar.

Select "New VI" to create a new VI or "Open VI" if you have an existing VI that you want to work on.

% Front Panel Design

The front panel is where you design the user interface of your VI. You can add controls (inputs) and indicators (outputs) by using the tools in the "Controls Palette" on the left-hand side of the LabVIEW window.

Drag and drop controls (e.g., buttons, knobs, numeric inputs) onto the front panel to specify user inputs.

Drag and drop indicators (e.g., graphs, numeric displays) onto the front panel to display the results or data.

Arrange and customize the front panel elements to create a user-friendly interface for your VI.

% Block Diagram Design

The block diagram is where you define the functionality of your VI by connecting wires between the controls and indicators and adding programming logic.

Right-click on the block diagram and use the "Functions Palette" on the left to add functions and structures.

Use wires to connect the controls to functions and indicators to display the results. Write LabVIEW code to perform the desired operations, calculations, or data processing.

% Wiring and Programming

Wire controls and indicators together to establish data flow between the front panel and the block diagram. To wire, click and drag from a control to an indicator or to a function's input.

Write LabVIEW code on the block diagram using graphical programming. You can use structures like loops and case structures for more complex operations.

% Save the VI

Click on "File" and select "Save" to save your VI with a meaningful name and location on your computer.

% Run the VI

Click the "Run" button on the toolbar to execute your VI. Interact with the controls on the front panel and observe the results on the indicators.

% Debug and Test

LabVIEW provides debugging tools such as probes, highlight execution, and breakpoints to help you identify and resolve issues in your VI.

% Documentation and Deployment

Consider documenting your VI with comments and descriptions to make it more understandable for others. You can also create a standalone application or share the VI with others by creating an executable file or a LabVIEW run-time engine package.

Try:

Problem 1: Understanding the Existing VI

1. Download the provided LabVIEW VI file (e.g., "Assignment_VI.vi") or use an existing VI if available.
2. Open the VI in LabVIEW.
3. Examine the front panel and block diagram of the VI to understand its functionality and user interface.

Problem 2: Editing and Building the Existing VI

1. Edit the existing VI to achieve the following:
 - Modify the user interface by adding at least one additional control (e.g., a numeric input, a switch, a slider).
 - Modify the VI's functionality by adding LabVIEW code to perform a specific operation (e.g., mathematical calculations, data processing).
2. Build and run the edited VI to ensure it works as intended.

Problem 3: Creating a Sub VI

1. Identify a portion of the LabVIEW code in the edited VI that can be modularized into a separate sub VI. This portion should be a self-contained task or operation.
2. Create a new VI (Sub VI) that performs the identified task or operation. This sub VI should have inputs and outputs as needed.
3. Incorporate the Sub VI into the edited VI by replacing the relevant LabVIEW code with a call to the Sub VI.
4. Build and run the edited VI with the Sub VI to verify that the modularization works correctly.

Problem 4: Documentation and Presentation

1. Document your changes and explain the functionality of the edited VI and the purpose of the Sub VI in a clear and concise report.
2. Create a brief presentation (e.g., slides or a video) demonstrating the edited VI, the Sub VI, and the modifications you made. Explain the benefits of modularization using Sub VIs.

Submission Instructions:

1. Submit the edited LabVIEW VI file (with your modifications).
2. Submit the created Sub VI as a separate LabVIEW file.
3. Submit a report (in PDF or Word format) that includes documentation of your changes and explanations.
4. Provide a link or file for your presentation materials.

8. Generation of Common Wave Forms Using LabVIEW

Generating common waveforms using LabVIEW is relatively straightforward with the built-in functions and structures provided by the LabVIEW programming environment. You can create waveforms such as sine waves, square waves, triangular waves, and saw tooth waves using LabVIEW. Below are the steps and syntax to generate some of these common waveforms:

Hints

Generating a Sine Wave:

1. Place a while loop on the block diagram to continuously generate the sine wave.
2. Inside the loop, use a Numeric Control to adjust the frequency of the sine wave.
3. Use a Sine Waveform function to generate the sine wave.

```
/*      Generating a Sine Wave:*/
```

```
ne Waveform VI -> Amplitude (constant) = 1.0 -> Frequency (control) =  
[frequencycontrol value]
```


Generating a Square Wave:

1. Place a while loop on the block diagram.
2. Inside the loop, use a Numeric Control to adjust the frequency of the square wave.
3. Use a Square Waveform function to generate the square wave.

```
/*      Generating a Square Wave: */  
  
Square Waveform VI -> Amplitude (constant) = 1.0 -> Frequency (control) =  
[frequencycontrol value]
```

Generating a Triangular Wave:

1. Place a while loop on the block diagram.
2. Inside the loop, use a Numeric Control to adjust the frequency of the triangular wave.
3. Use a Triangular Waveform function to generate the triangular wave.

```
/*      Generating a Triangulate Wave: */  
  
Triangular Waveform VI -> Amplitude (constant) = 1.0 -> Frequency (control)  
=[frequency control value]
```

Generating a Saw tooth Wave:

1. Place a while loop on the block diagram.
2. Inside the loop, use a Numeric Control to adjust the frequency of the saw tooth wave.
3. Use a Saw tooth Waveform function to generate the saw tooth wave.

```
/**      Generating a Saw Tooth Wave: */  
  
Saw tooth Waveform VI -> Amplitude (constant) = 1.0 -> Frequency (control)  
=[frequency control value]
```

Try:

1. Display of maximum and minimum values of a sinusoidal signal.
2. Display of modulation of two sinusoidal signals.

10. Frequency measurement using Lissajous pattern in LabView

Frequency measurement using Lissajous pattern in LabVIEW involves the use of an oscilloscope or waveform display to analyze the Lissajous pattern formed when two sinusoidal signals with different frequencies are plotted against each other. The frequency measurement can be performed by counting the number of intersections or lobes of the Lissajous pattern within a known time period. Below is a step-by-step guide on how to implement frequency measurement using Lissajous pattern in LabVIEW:

Hints

/ Create a LabVIEW VI*/**

Open LabVIEW and create a new VI (Virtual Instrument).

/ Generate Lissajous Figures*/**

Generate two sinusoidal signals with different frequencies and amplitudes using appropriate Signal Generation functions. Use separate Frequency controls for each signal. Create two separate Waveform Charts or Graphs on the front panel. Wire the generated signals to the Waveform Charts or Graphs.

/ Set Up the Timebase */**

Add a Numeric Control for the timebase (the time interval over which you want to measure frequency).

Connect the Numeric Control to a Wait (ms) function to introduce a delay in your loop.

/ Count Lissajous Lobe Intersections */**

Place a While Loop on the block diagram. Inside the loop, increment a counter (initialized to zero) every time you detect an intersection or lobe crossing of the Lissajous figure. Compare the elapsed time (using the Tick Count function) with the desired timebase, and exit the loop when the desired time has passed.

/ Calculate Frequency */**

Outside the loop, calculate the frequency using the following formula: $\text{Frequency (Hz)} = \frac{\text{Number of Intersections}}{\text{Timebase (seconds)}}$

/ Display the Frequency */**

Create a Numeric Indicator to display the calculated frequency. Connect the calculated frequency to the Numeric Indicator.

/ Run the VI */**

Run the VI and observe the Lissajous figure on the waveform charts. The calculated frequency will be displayed on the Numeric Indicator when the VI completes.

Try:

1. Display of Lissajous pattern for sinusoidal voltages of same frequencies.
2. Display of Lissajous pattern for sinusoidal voltages of same different frequencies.

11. Structures Using LabVIEW

Analyze the virtual instrumentation (VI) using control loops, arrays, charts and graphs in LabVIEW software

Hints

/ Open LabVIEW and create a new VI */**

/ Place the following controls on the front panel */**

- Numeric Control for Frequency (Hz)
- Numeric Control for Amplitude (V)
- Numeric Control for Duration (seconds)
- Waveform Chart (for displaying the original signal)
- Waveform Graph (for displaying the FFT result)

/ Create a FOR loop on the block diagram with the following elements inside it */**

- Numeric Constant (0) wired to the FOR loop iteration terminal
- Numeric Control for Duration wired to the count terminal of the FOR loop

/ Inside the FOR loop, generate a sinusoidal signal using the Sine Waveformfunction */**

- Place a Sine Waveform function inside the FOR loop
- Wire the Frequency control to the "Frequency (Hz)" input of the Sine Waveformfunction
- Wire the Amplitude control to the "Amplitude (V)" input of the Sine Waveform function
- Use the FOR loop iteration terminal to generate a time signal (0, 1 sampleperiod, 2 sample periods, etc.)
- Wire the time signal to the "Time (s)" input of the Sine Waveform function

/ Wire the output of the Sine Waveform function to a Build Array function to create an array of the generated signal */**

/ After the FOR loop, add a Peak Detector VI to detect peaks in the signal */**

- Place a Peak Detector VI on the block diagram
- Wire the output array to the "Input Array" input of the Peak Detector
- Set the "Threshold" parameter of the Peak Detector VI to a suitable value

/ Connect the output of the Peak Detector VI to the "Data" input of the WaveformChart to display the original signal with peaks highlighted */**

/ Place a Waveform Chart Clear function outside the FOR loop and wire it to the chart to clear the chart before displaying the new data */**

/ Create a WHILE loop with the following elements inside it */**

- A Waveform Graph Clear function to clear the graph for displaying FFT results
- An FFT function to compute the Fast Fourier Transform of the signal:
- Wire the output array (from the Peak Detector) to the "Time Domain Signal" input of the FFT function
- A waveform chart to display the FFT result:
- Wire the output of the FFT function to the "Amplitude Spectrum" input of the waveform chart

/ Wire the output of the Waveform Chart Clear function inside the WHILE loop to clear the chart before displaying the new data */**

/ Run the VI to generate the sinusoidal signal, detect peaks, compute the FFT, and display the results on the charts */**

/ Analyze the displayed results, including peak positions and frequency components in the FFT result */**

This LabVIEW example demonstrates the use of FOR and WHILE loops, charts (waveform chart and waveform graph), arrays, and analysis VIs (Peak Detector and FFT) in a signal processing application. Students can modify and extend this example to explore more complex signal processing and analysis tasks.

Try:

1. Obtain VI using for loop and While loop.
2. Obtain VI using charts and arrays.
3. Obtain VI using graphs.

12. Simulation of low pass and high pass filters using digital simulation

To plot the characteristics of low-pass and high-pass filters using MATLAB, you can follow these steps and use the **'freqz'** function for plotting the frequency response:

Hints;

```
/** Design the Filters */
```

Design low-pass and high-pass filters using the 'fir1' or 'fdesign' functions. Here's an example using 'fir1':

```
% Design a low-pass filter
lowpass_order = 50; % Filter order
lowpass_cutoff_frequency = 0.2; % Cutoff frequency (normalized)
lowpass_filter = fir1(lowpass_order, lowpass_cutoff_frequency);

% Design a high-pass filter
highpass_order = 50; % Filter order
highpass_cutoff_frequency = 0.2; % Cutoff frequency (normalized)
highpass_filter = fir1(highpass_order, highpass_cutoff_frequency, 'high');
```

```
/** Plot the Frequency Response */
```

Plot the frequency response of the filters using the 'freqz' function:

```
% Frequency response of the low-pass filter
figure;
freqz (lowpass_filter, 1, 1024);
title ('Low-Pass Filter Frequency Response');
xlabel ('Normalized Frequency (\pi radians/sample)');
ylabel ('Magnitude (dB)');

% Frequency response of the high-pass filter
figure;
freqz (highpass_filter, 1, 1024);
title ('High-Pass Filter Frequency Response');
xlabel ('Normalized Frequency (\pi radians/sample)');
ylabel ('Magnitude (dB)');
```

In this code:

'freqz (lowpass_filter, 1, 1024)' calculates and plots the frequency response of the low-pass filter.

'freqz (highpass_filter, 1, 1024)' calculates and plots the frequency response of the high-pass filter.

```
/** Display the Phase Response (Optional) */
```

If you also want to display the phase response, you can modify the 'freqz' function as follows:

```
% Phase response of the low-pass filter
figure;
[h, w] = freqz (lowpass_filter, 1, 1024);
plot (w, unwrap(angle(h)));
title ('Low-Pass Filter Phase Response');
xlabel ('Normalized Frequency (\pi radians/sample)');
ylabel ('Phase (radians)');

% Phase response of the high-pass filter
figure;
[h, w] = freqz (highpass_filter, 1, 1024);
plot (w, unwrap(angle(h)));
title ('High-Pass Filter Phase Response');
xlabel ('Normalized Frequency (\pi radians/sample)');
ylabel ('Phase (radians)');
```

Try:

Problem 1: Filter Design

1. Design a low-pass filter with the following specifications:
 - Filter Order: 40
 - Cutoff Frequency: 0.2 (normalized frequency)
2. Design a high-pass filter with the following specifications:
 - Filter Order: 30
 - Cutoff Frequency: 0.3 (normalized frequency)

Problem 2: Frequency Response Plot

1. Plot the magnitude frequency response (in dB) of both the low-pass and high-pass filters.
 - Use the **'freqz'** function to calculate and plot the frequency response.
 - Label the x-axis as "Normalized Frequency (π radians/sample)" and the y-axis as "Magnitude (dB)."
2. Plot the phase frequency response (in radians) of both filters.
 - Use the **freqz** function to calculate and plot the phase response.
 - Label the x-axis as "Normalized Frequency (π radians/sample)" and the y-axis as "Phase (radians)."

Problem 3: Filtered Signal

1. Generate a noisy input signal using MATLAB. You can use the **'randn'** function to generate random noise and the **'sin'** function to generate a sinusoidal signal.
2. Apply both the low-pass and high-pass filters to the noisy signal to obtain filtered signals.
3. Plot the noisy input signal, the low-pass filtered signal, and the high-pass filtered signal on the same graph.

Problem 4: Frequency Response Analysis

1. Analyze the frequency response plots and explain how the magnitude and phase characteristics of the filters align with their designed specifications
2. Describe the differences in filtering results between the low-pass and high-pass filters. Discuss which frequencies are attenuated and which are passed through.

Problem 5: Submission

1. Prepare a report summarizing your filter design, frequency response analysis, and filtered signal results.
2. Include the MATLAB code used for filter design, frequency response plotting, and signal processing in your report.
3. Provide explanations and interpretations of the results in the report.

12. Sensor Circuit Using LAB View

Designing the electric and electronic circuit of a sensor in LabVIEW involves creating a virtual representation of the circuit using LabVIEW's graphical programming interface. You'll use various components, such as controls, indicators, and wiring, to simulate the behavior of the physical sensor circuit. Below is a simplified example of how you can design a virtual sensor circuit in LabVIEW.

HINTS

/ Open LabVIEW and Create a New VI */**

Launch LabVIEW and create a new VI (Virtual Instrument).

/ Design the Front Panel */**

Add controls and indicators to the front panel to represent components in your sensor circuit. For example:

Numeric Controls for resistors, capacitors, or other components with adjustable values.

Switches or buttons to simulate the opening or closing of switches.

Numeric Indicators or Waveform Charts to display sensor readings.

You can find these controls and indicators in the "Controls" and "Indicators" palettes.

/ Create the Block Diagram */**

Switch to the block diagram by clicking on the "Block Diagram" tab.

/ Wire Components Together */**

Use wiring to connect the controls and indicators to simulate connections in your sensor circuit.

For example, use wires to connect a voltage source (control) to a resistor (indicator).

/ Implement Logic */**

Use LabVIEW's graphical programming to implement the logic and calculations needed

for your sensor circuit. You can use functions and structures like loops, case structures, and math functions.

/ Simulate Sensor Behavior */**

Implement code that simulates the behavior of the sensor. For instance, if you're simulating a temperature sensor, write code to generate temperature readings based on user inputs or predefined conditions.

/ Display Sensor Output */**

Use indicators (such as numeric indicators or waveform charts) to display the simulated sensor readings on the front panel.

/ Run the VI */**

Save and run your VI to see the simulated behavior of the sensor circuit.

Adjust control values on the front panel to see how they affect the sensor output.

/ Analyze and Validate */**

Analyze the behavior of your virtual sensor circuit to ensure it behaves as expected.

Validate the circuit's response to different inputs and conditions.

/ Document Your Design*/**

Document your virtual sensor circuit design, including circuit diagrams, code explanations, and any assumptions you've made.

/ Extend and Customize */**

Depending on your needs, you can extend the virtual sensor circuit to include more components, sensors, or complex behavior.

Try:

Problem 1: Front Panel Design

1. Open LabVIEW and create a new VI.
2. On the front panel, design the user interface to represent an electric and electronic circuit of a sensor. Include the following elements:
 - Numeric Controls for adjusting resistor values (e.g., R1, R2).
 - Toggle Switches to simulate the opening and closing of switches.
 - Numeric Indicators to display voltage, current, or other relevant values.
 - A Waveform Chart to display dynamic sensor data.
 - Any additional elements you consider relevant to your sensor circuit.

Problem 2: Block Diagram Implementation

1. Switch to the block diagram and use LabVIEW's graphical programming to simulate the behavior of the sensor circuit. Implement the following:
 - Wiring to connect the components as per the designed circuit.
 - Logic for calculating voltages, currents, or sensor readings based on component values and switch positions.
 - Simulation of time-dependent behavior if applicable (e.g., transient response).

Problem 3: Simulate Sensor Operation

1. Create a scenario where the sensor circuit interacts with an external environment or input signal. For example, if simulating a temperature sensor, simulate temperature changes over time.
2. Implement code that calculates and updates sensor data based on the interaction with the environment or input signal.

Problem 4: Real-Time Monitoring

1. Configure LabVIEW to provide real-time monitoring of sensor data.
 - Use a loop structure to continuously update sensor readings.
 - Display the sensor data in real-time on the Waveform Chart on the front panel.

Problem 5: Data Analysis and Validation

1. Run the VI and observe the behavior of the virtual sensor circuit.
 - Verify that the circuit responds correctly to changes in component values and switch positions.
 - Analyze the data displayed on the Waveform Chart and ensure it matches the expected behavior.

Problem 6: Report and Documentation

1. Prepare a report that includes the following:
 - A description of the designed sensor circuit.
 - A discussion of how LabVIEW was used to simulate its behavior.
 - Results and observations from running the VI.
 - An analysis of the data collected during the simulation.
2. Include screenshots of the LabVIEW front panel and block diagram in your report.

Problem 7: Presentation

Prepare a brief presentation to demonstrate your virtual sensor circuit and discuss your findings.

12. Proximity Sensor Circuit Using LAB View

Measuring the speed of a machine using a proximity sensor in LabVIEW typically involves using a sensor to detect the position of a rotating object and then calculating the speed based on the change in position over time.

HINTS

/ Hardware Setup */**

Connect your proximity sensor to the appropriate data acquisition hardware (e.g., a DAQ card) and ensure that it is properly configured.

/ Open LabVIEW and Create a New VI */**

Launch LabVIEW and create a new VI (Virtual Instrument).

/ Front Panel Design */**

On the front panel, add the necessary controls and indicators:

- Numeric Indicator for displaying the speed.
- A Start Button to initiate speed measurement.
- A Graph or Chart for visualizing speed data (optional).
- Any other controls or indicators you may need.

/ Create the Block Diagram */**

Switch to the block diagram by clicking on the "Block Diagram" tab.

/ Initialize Variables */**

Use a Numeric control to set the initial position of the machine.

/ While Loop for Measurement */**

Place a While Loop on the block diagram to continuously measure and update the speed.

Use a Case Structure with the Stop Button to control the loop's execution.

/ Read Proximity Sensor Data */**

Inside the While Loop, use DAQ functions or appropriate libraries to read data from the proximity sensor.

Store the sensor's position data in a variable.

/ Calculate Speed */**

Calculate the speed of the machine by determining the change in position over time (velocity).

You can use LabVIEW's Math functions to perform this calculation.

```
/** Display Speed */
```

Display the calculated speed on the Numeric Indicator on the front panel.

```
/** Optional: Data Visualization */
```

If desired, you can also plot the speed data on a Graph or Chart in real-time to visualize the machine's speed profile.

```
/** Start and Stop Controls */
```

Use the Start Button to initiate speed measurement by enabling the While Loop.

Use the Stop Button to stop the measurement by disabling the While Loop.

```
/** Run the VI */
```

Save and run your VI to measure and display the speed of the machine.

This is a simplified example, and in practice, you may need to consider additional factors such as calibration, noise filtering, and error handling. The specific implementation details can vary depending on the type of proximity sensor and data acquisition hardware you are using.

Try:

Problem 1: Hardware Setup

Set up a proximity sensor and connect it to a DAQ (Data Acquisition) device or hardware.

Problem 2: LabVIEW VI Design

1. Open LabVIEW and create a new VI.
2. Design the front panel with the following elements:
 - A numeric indicator to display the measured speed (in RPM or other appropriate units).
 - A start button to initiate speed measurement.
 - A stop button to stop the measurement.
 - A chart or graph indicator to display the speed profile (optional but recommended).
 - Any additional controls or indicators you consider necessary.

Problem 3: Block Diagram Implementation

1. Create a While Loop on the block diagram.
2. Use a Case Structure inside the loop to handle the start and stop buttons. The loop should only execute when the start button is pressed and stop when the stop button is pressed.
3. Use DAQ functions or libraries (appropriate for your hardware) to read data from the proximity sensor. You may need to configure the DAQ task or device settings.
4. Calculate the speed of the machine based on the sensor data. Consider the appropriate formula for converting sensor data into speed (e.g., RPM calculation).

5. Display the calculated speed on the numeric indicator on the front panel.
6. If desired, plot the speed data on the chart or graph indicator for real-time visualization.

Problem 4: User Interface

1. Test the VI by running it.
 - Click the start button to initiate speed measurement.
 - Observe the speed reading on the numeric indicator.
 - Use the stop button to halt the measurement.

Problem 5: Data Analysis

1. Analyze the data collected during the measurement.
 - Calculate the average speed.
 - Identify any variations or anomalies in the speed profile.

Problem 6: Report

1. Prepare a report summarizing your LabVIEW program's design and functionality.
 - Include screenshots of the LabVIEW front panel and block diagram.
 - Describe the hardware setup and connection.

Problem 7: Presentation

1. Create a brief presentation to demonstrate your LabVIEW program and discuss your findings.
 - Discuss how the program measures machine speed.

V. REFERENCE BOOKS:

1. A Chakrabarti, "Circuit Theory", Dhanpat Rai Publications, 6th edition, 2006.
2. William Hayt, Jack E Kemmerly S.M. Durbin, "Engineering Circuit Analysis", Tata McGraw Hill, 7th edition, 2010.
3. K S Suresh Kumar, "Electric Circuit Analysis", Pearson Education, 1st edition, 2013.

VI. ELECTRONIC RESOURCES:

1. <https://www.ee.iitkgp.ac.in>
2. <https://www.iare.ac.in/>

VIII. MATERIALS ONLINE:

1. Course template
2. Lab Manual



INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal - 500 043, Hyderabad, Telangana

COURSE CONTENT

DC MACHINES LABORATORY								
III Semester: EEE								
CourseCode	Category	Hours/Week			Credits	MaximumMarks		
AEED09	Core	L	T	P	C	CIA	SEE	Total
		0	0	2	1	40	60	100
Contact Classes: Nil	Tutorial Classes: Nil	Practical Classes: 45				Total Classes: 45		
Prerequisite: Electronic Devices and Circuits Laboratory								

I. COURSE OVERVIEW:

This laboratory course is to meet the requirements of practical work meant for basic operation, analysis and design of electrical machines. It provides hands-on experience by examining the electrical and mechanical characteristics of various DC machines. Analyze the characteristics of DC machines and separate the various losses in electrical machines by conducting different tests.

II. COURSES OBJECTIVES:

The students will try to learn

- The elementary experimental and modelling skills for handling problems with electrical machines in the industries and domestic applications to excel in professional career.
- The operation of DC Machines and its role in power transmission and distribution.
- The intuitive knowledge needed to test and analyze the performance leading to design of electric machines by conducting various tests and calculate the performance parameters.

III. COURSE OUTCOMES:

At the end of the course students should be able to:

- CO 1 Analyze the performance characteristics of dc machine under various loading conditions.
- CO 2 Determine the critical field resistance and speed of dc shunt generator using open circuit characteristics.
- CO 3 Examine the performance of DC shunt machine with different speed control techniques and predetermine the efficiency.
- CO 4 Estimate and separate the core losses in dc machine by conducting a suitable test.
- CO 5 Examine the performance and speed control of dc machines using simulation tools.

IV. COURSE CONTENT:

EXERCISES FOR ELECTRICAL DC MACHINES LABORATORY

1. OPEN CIRCUIT CHARACTERISTICS OF DC SHUNT GENERATOR.

Develop the circuit for analyzing the characteristics of DC shunt generator.

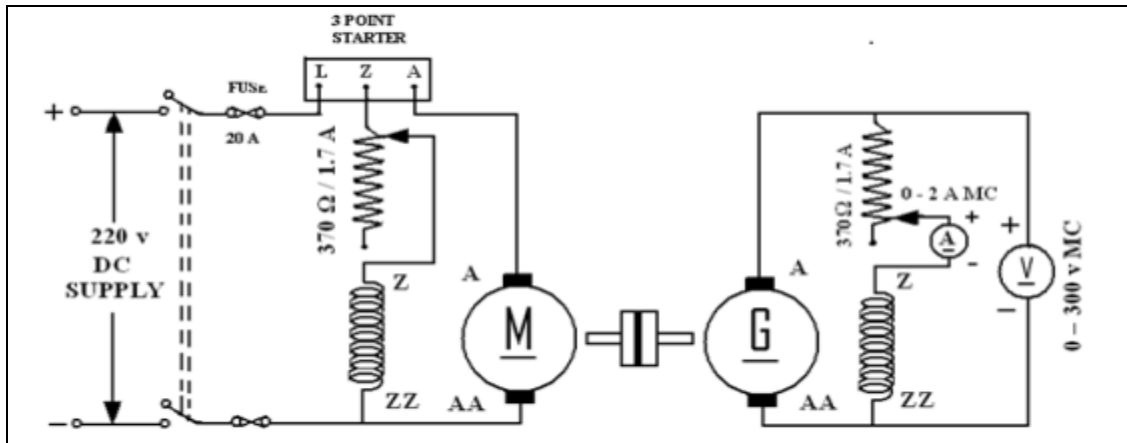


Figure 1 – Open Circuit Characteristics of DC Shunt Generator

Try

1. From the Open circuit characteristics calculate the critical resistance of field winding.
2. Using magnetization characteristics calculate the critical speed of DC shunt generator at 100 ohms in Figure 1.
3. Determine the performance of DC generator using the magnetization curve.
4. Calculate the critical value of shunt field resistance at 1500 rpm.

2. LOAD TEST ON DC SHUNT GENERATOR

Design the DC shunt generator circuit under various load conditions for analyzing the performance of the machine

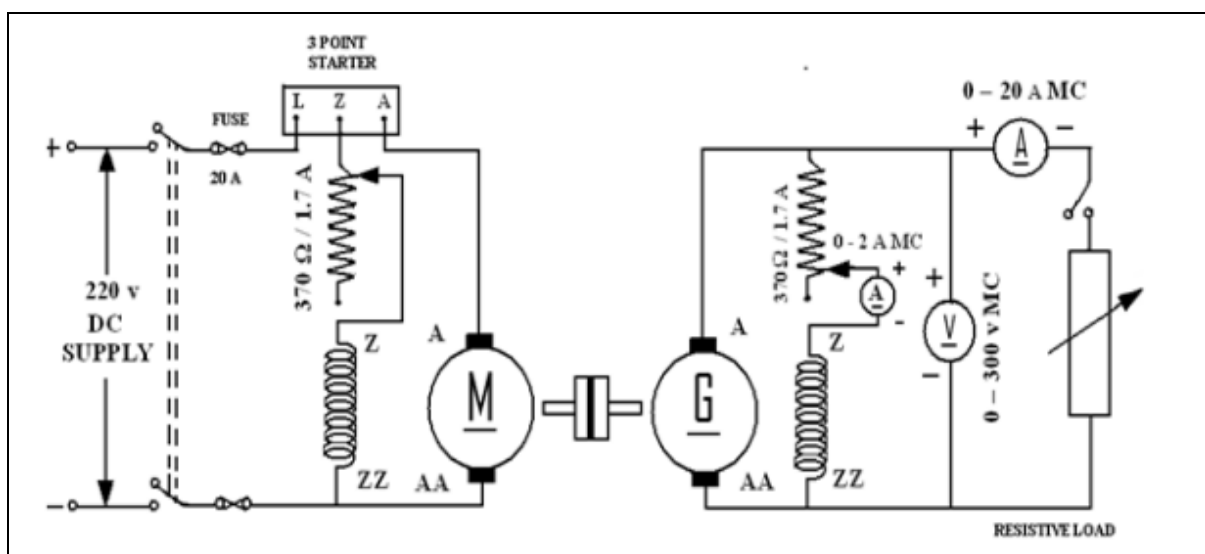


Figure 2 – Load Test on D.C Shunt Generator

Try

1. Calculate the different armature currents (i.e. I_a equal I_L plus I_f) for shunt generator under various loads in Figure 2.
2. Draw the External characteristics using armature currents (0 to 13 A) with respect to the Load currents.
3. Draw the Internal characteristics using generated induced emf (0 to 220V) with respect to the field currents (0 to 2A).

3. LOAD TEST ON DC SERIES GENERATOR

Design the DC series generator circuit under various load conditions for analyzing the performance of the machine.

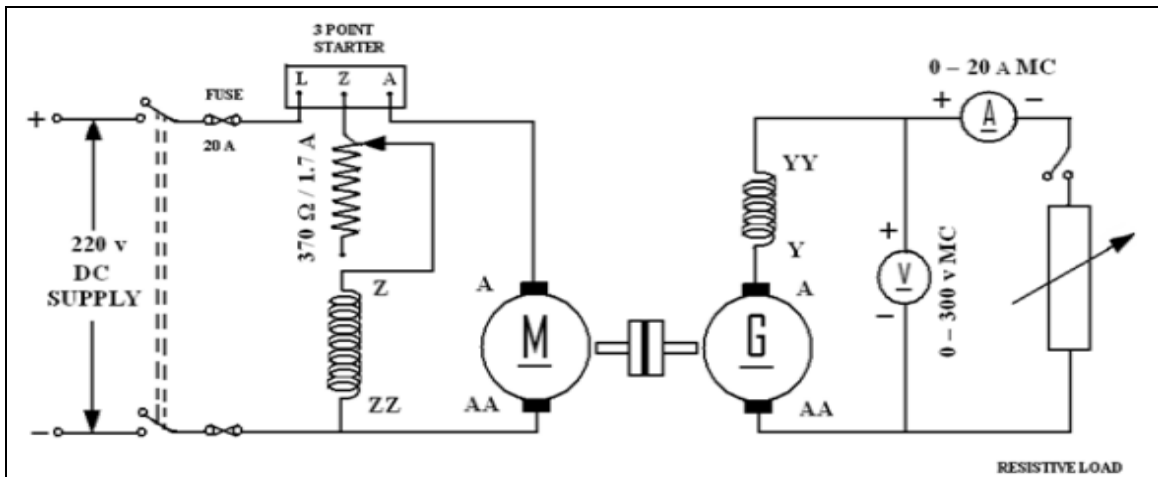


Figure 3 – Load Test on D.C Series Generator

Try

1. Calculate the different armature currents (i.e. I_a equal I_L plus I_f) for series generator under various Loads In Figure 3.
2. Draw the External characteristics using armature currents (0 to 13 A) with respect to the load currents.
3. Draw the Internal characteristics using generated induced emf (0 to 220V) with respect to the field currents (0 to 2A).

4. LOAD TEST ON DC COMPOUND GENERATOR

Design the DC compound generator circuit under various load conditions for analyzing the performance of the machine.

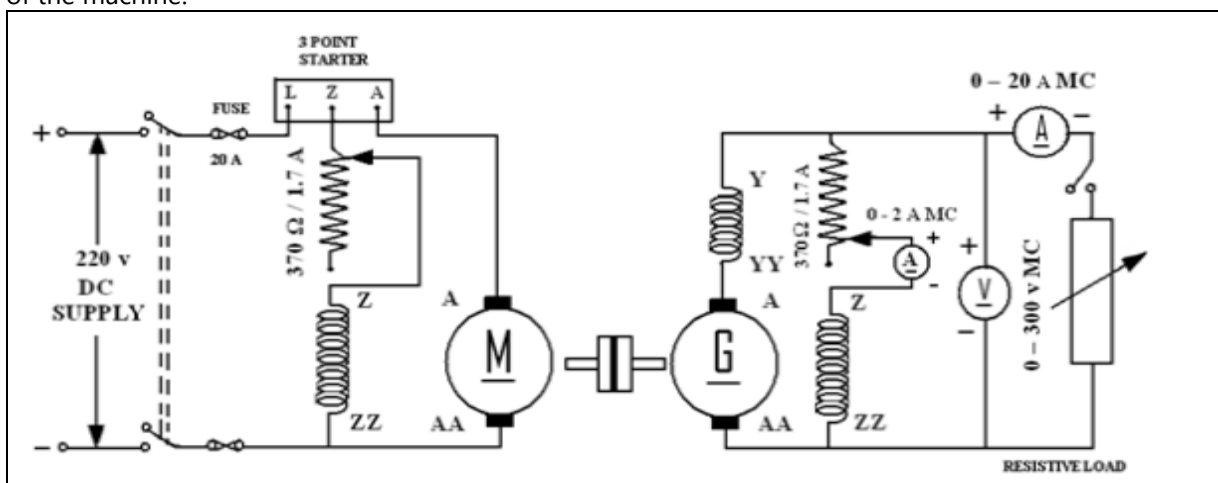


Figure 4 – Load Test on D.C Compound Generator

Try

1. Calculate the different armature currents (i.e. I_a equal I_L plus I_f) for compound generator

under various loads in Figure 4.

2. Draw the External characteristics using armature currents (0 to 13 A) with respect to the load currents.
3. Draw the Internal characteristics using generated induced emf (0 to 220V) with respect to the field currents (0 to 2A).

5. HOPKINSON'S TEST ON TWO IDENTICAL D.C SHUNT MACHINES

Develop a method of testing for two identical dc shunt machines which are mechanically coupled and also electrically connected in parallel.

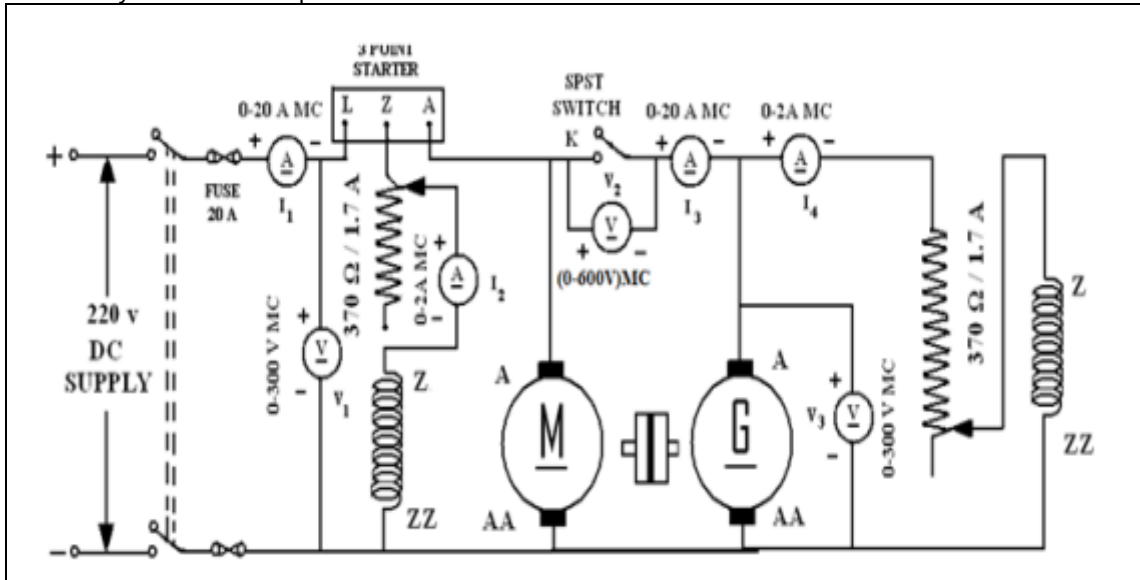


Figure 5 - Identical DC Shunt Machines

Try

1. Calculate the efficiency of two identical dc shunt machines if the armature resistance of each machine is 0.025 ohms, line voltage of 230V and line current excluding both the field currents 2A,
2. Motor armature current 10A, field current 1A and 2A Figure 5.
3. Draw the performance characteristics of two identical dc shunt machines.
4. Find the iron losses depend on the emf generated in the armature.

6. FIELD'S TEST ON TWO IDENTICAL D.C SERIES MOTORS

1. Develop a method of testing for two similar dc series machines depend on the accuracy with which the motor input and generator output are measured.
2. Determine load or unknown current through a R_4 resistor using Thevenin's equivalent circuit.

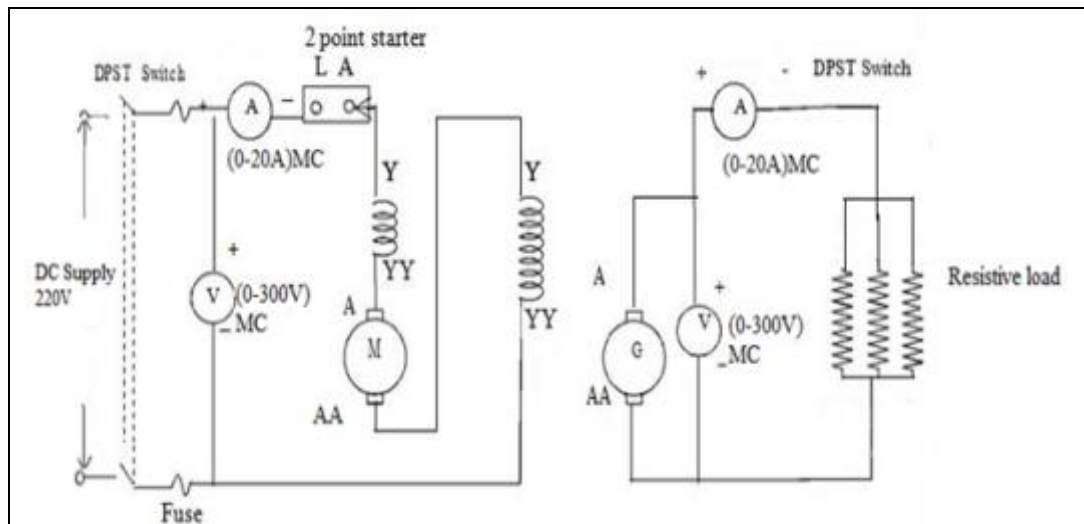


Figure 6 - Identical DC Shunt Machines

Try

1. Calculate the efficiency of both dc series machines if the armature resistance of each machine is 2 ohms in figure 6.
2. Draw the performance characteristics of two dc series machines.
3. Find the no load rotational losses of both the machines and total losses in the whole set.

7. SWINBURNE'S TEST AND SPEED CONTROL OF D.C SHUNT MOTOR

Design the suitable test under no load conditions to measure no load losses in DC shunt machines and speed control of DC shunt motor.

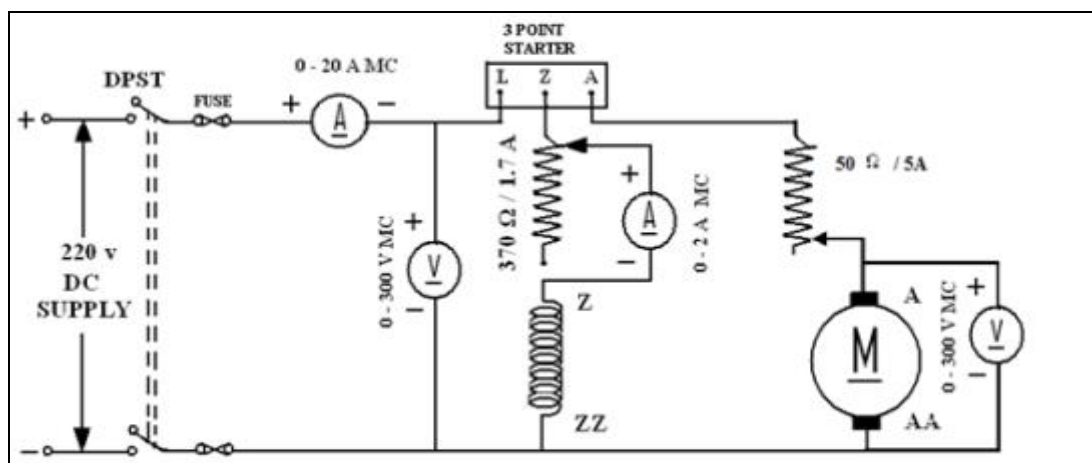


Figure 7 - Speed control of DC motor

Try

1. Calculate the output power and efficiency when motor takes 10A on full load and 5A on half Load in Figure 7.
2. Measure the no load machine losses by using indirect method of testing.
3. Perform the speed control by varying the armature circuit resistance and field circuit resistance of DC shunt motor.

8. BRAKE TEST ON DC COMPOUND MOTOR

Develop the circuit for conducting brake test on DC compound motor.

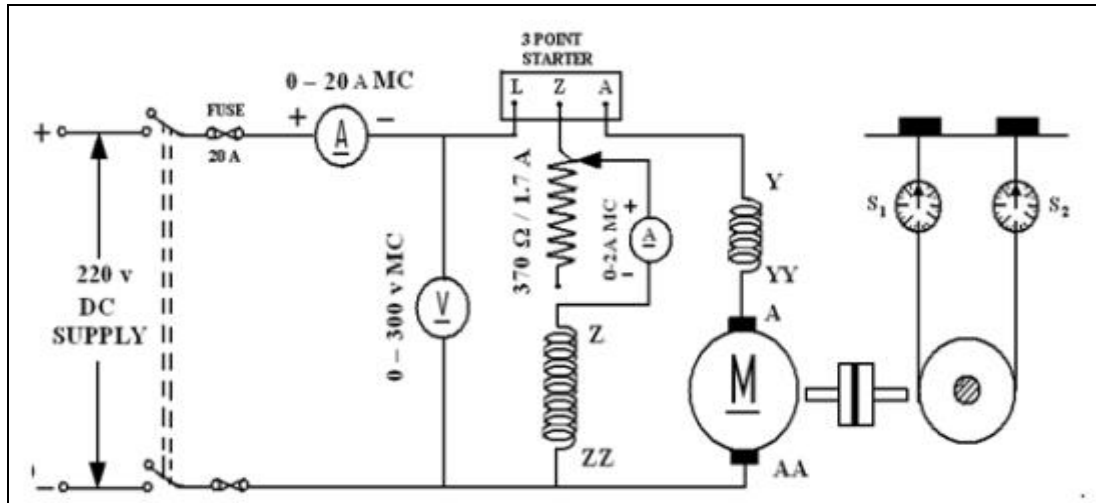


Figure 8 - Brake test on DC compound motor

Try

1. Calculate the efficiency of DC compound motor under different load conditions in Figure 8.
2. Calculate the shaft torque and shaft power at rated load.
3. Determine the mechanical output power under different weights.

9. BRAKE TEST ON DC SHUNT MOTOR

Develop the circuit for conducting brake test on DC shunt motor

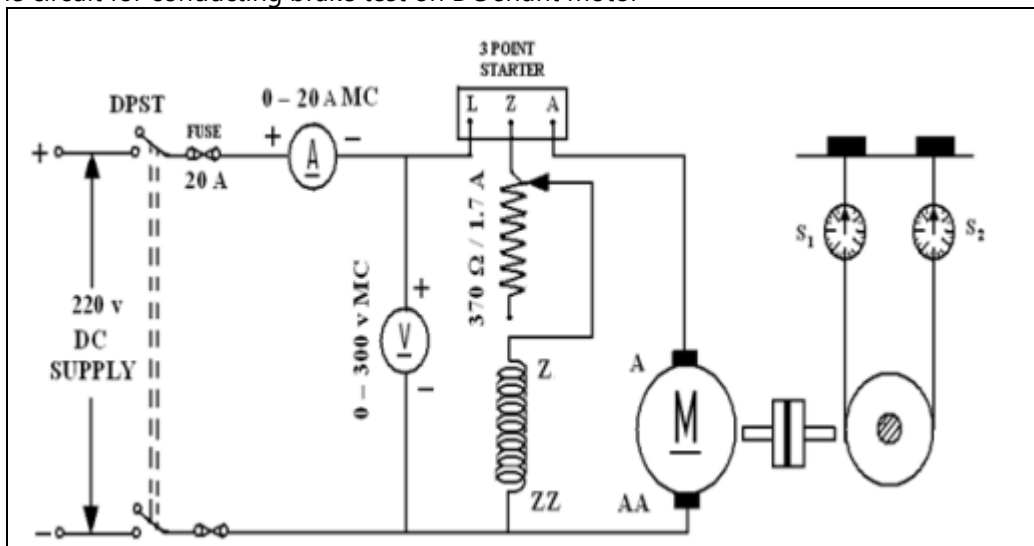


Figure 9 - Brake test on DC shunt motor

Try

1. Calculate the efficiency of DC shunt motor under different load conditions.
2. Calculate the shaft torque and shaft power at rated load.
3. Determine the mechanical output power under different weights

10. RETARDATION TEST

Develop the test for separating the mechanical losses of the DC shunt machine

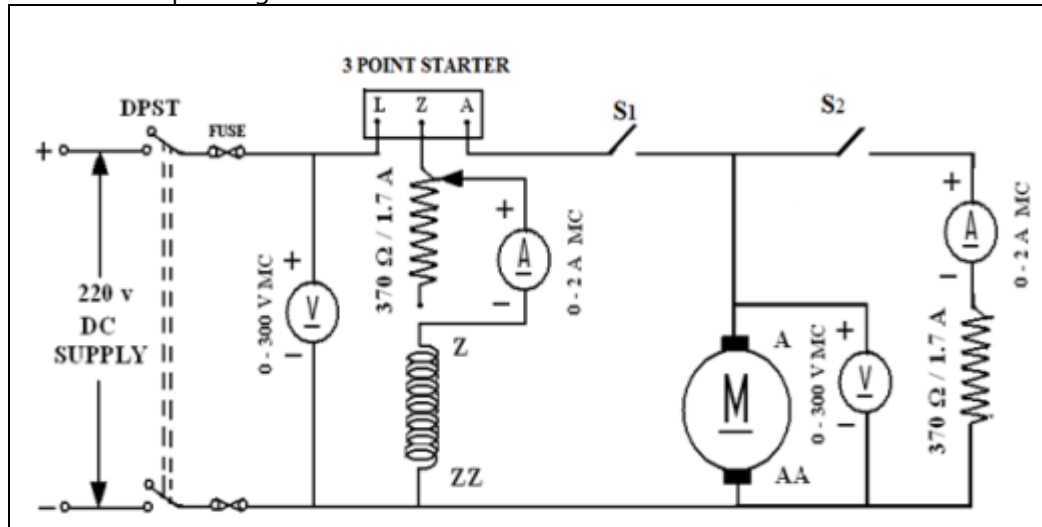


Figure 10 - Retardation Test

Try

1. Find the rotational losses i.e. friction, wind age and iron losses.
2. Measure the moment of inertia of the armature under normal speed of a DC machine 1000rpm.
3. Calculate the efficiency of DC shunt machine when time taken for the speed to fall from 1030 rpm to 970 rpm is 15 seconds with field normally excited in Figure 10.

11. SEPARATION OF LOSSES IN DC SHUNT MOTOR

Design the circuit for separating the iron losses in DC shunt motor.

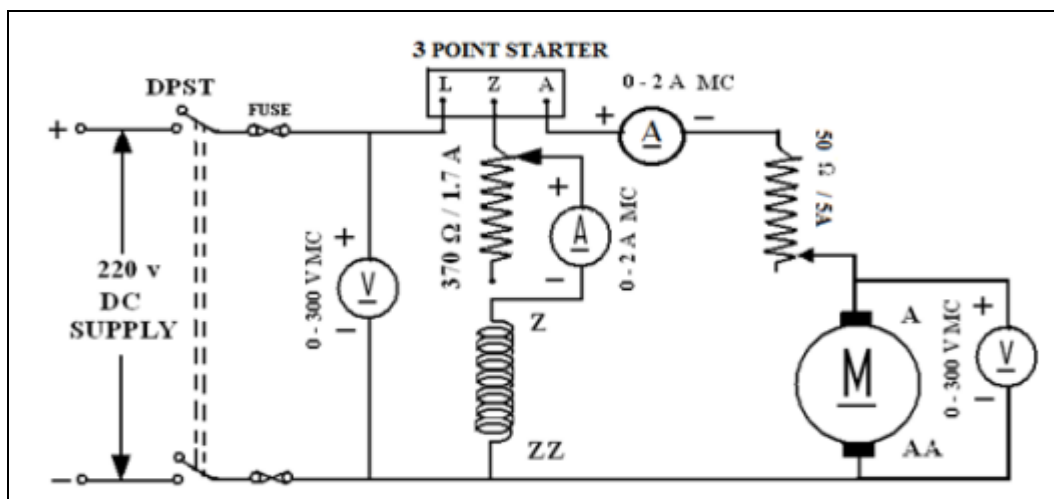


Figure 11 - Separation of losses in d.c shunt motor

Try

1. Calculate the hysteresis and eddy current losses in Dc shunt motor.
2. Draw the curve for total iron losses for various field currents.

12. MAGNETIZATION CHARACTERISTICS OF DC SHUNT GENERATOR USING DIGITAL SIMULATION

Develop the circuit for analyzing the magnetization characteristics of DC shunt generator using Digital simulation.

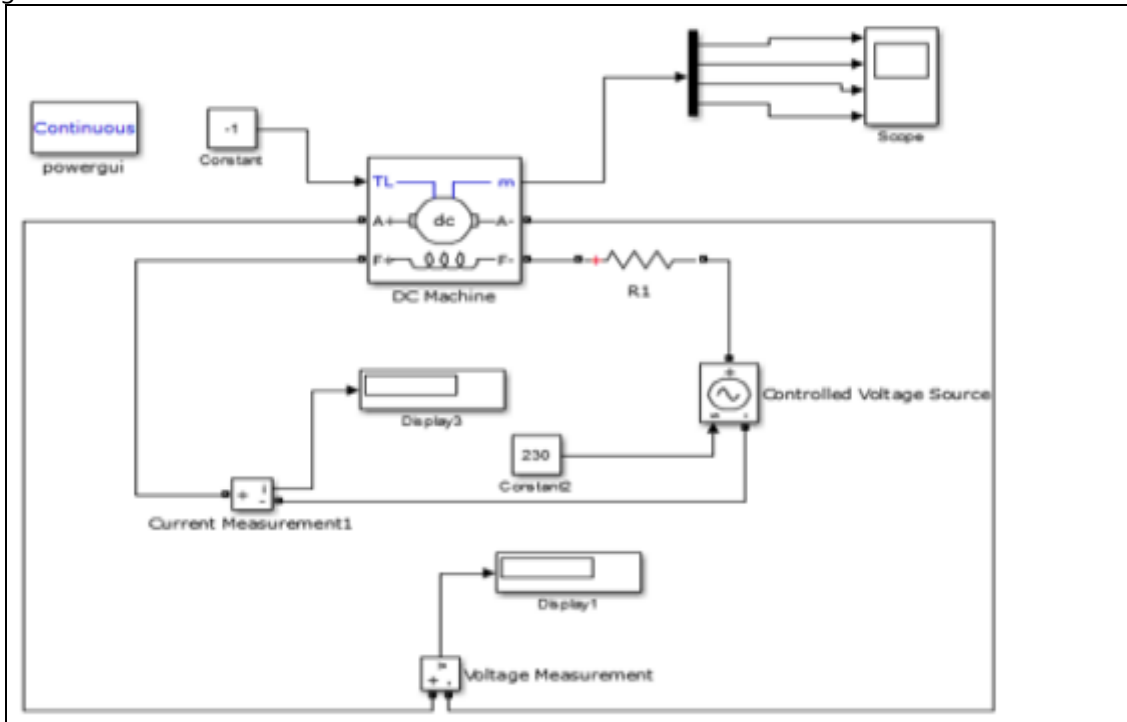


Figure – 12 Magnetization Characteristics of D.C Shunt Generator using Digital Simulation

Try

1. From the Open circuit characteristics calculate the critical resistance of field winding.
2. Using magnetization characteristics calculate the critical speed of DC shunt generator at 100 ohms.
3. Determine the performance of DC generator using the magnetization curve.
4. Calculate the critical value of shunt field resistance at 1500 rpm.

13. LOAD TEST ON DC SHUNT GENERATOR USING DIGITAL SIMULATION

Design the DC shunt generator circuit under full, 3/4th, half and 1/4th load conditions for analyzing the performance of the machine using Digital simulation.

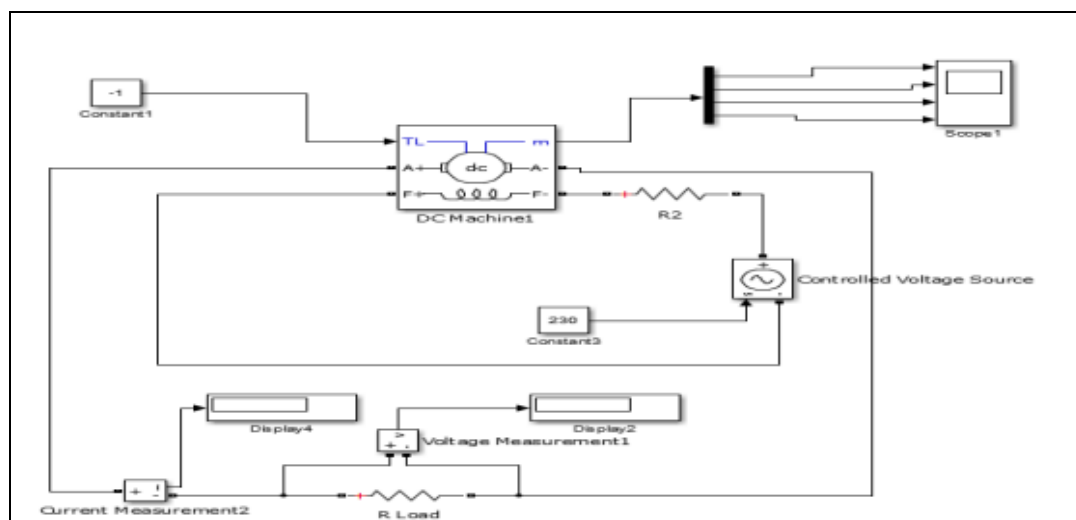


Figure – 13 Load Test on D.C Shunt Generator using Digital Simulation

Try

1. Calculate the different armature currents (i_a equal i_l plus i_f) for shunt generator under various loads.
2. Draw the External characteristics using armature currents (0 to 13 A) with respect to the load currents
3. Draw the Internal characteristics using generated induced emf (0 to 220V) with respect to the field currents (0 to 2A).

14. SPEED CONTROL OF D.C SHUNT MOTOR USING DIGITAL SIMULATION

Design the suitable test for speed control of DC shunt motor using MATLAB.

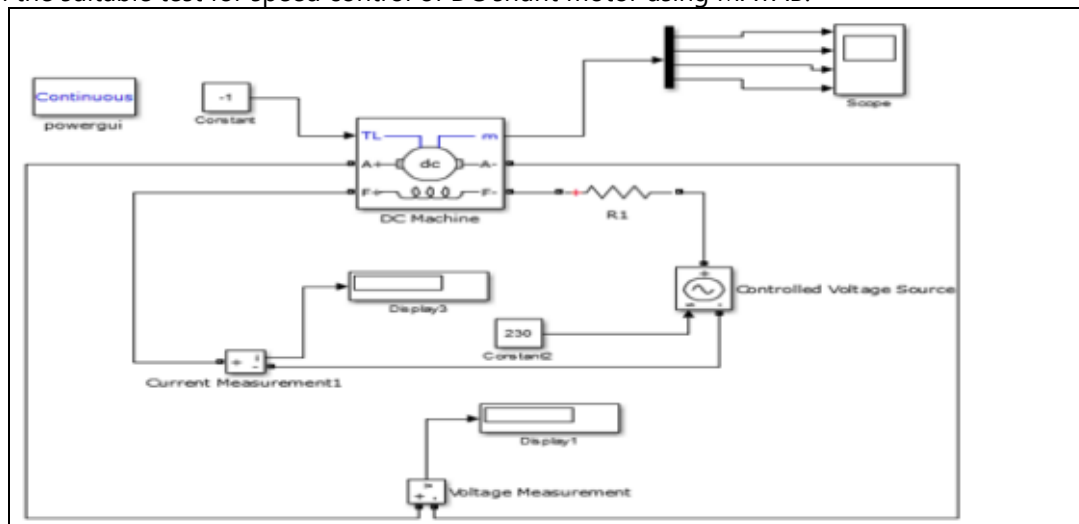


Figure – 14 speed control of D.C Shunt Motor using digital Simulation

Try

1. Perform the speed control by varying the armature circuit resistance of DC shunt motor.
2. Perform the speed control by varying the field circuit resistance of DC shunt motor.

V. TEXTBOOKS:

1. A Chakrabarti, “*Circuit Theory*”, Dhanpat Rai Publications, 8th edition, 2021.
2. William Hayt, Jack E Kemmerly S.M. Durbin, “*Engineering Circuit Analysis*”, Tata McGraw Hill, 9th edition, 2020.

VI. REFERENCE BOOKS:

1. CLWadhwa, *Electrical Circuit Analysis including Passive Network Synthesis*, International, 2nd Edition, 2009.
2. David A Bell, *Electric circuits*, Oxford University Press, 7th Edition, 2009.

VII. ELECTRONIC RESOURCES:

1. <https://www.allaboutcircuits.com/textbook/>
2. https://onlinecourses.nptel.ac.in/noc22_ee93/preview
3. <https://www.iare.ac.in>

VIII. MATERIALS ONLINE:

1. Course template
2. Lab manual



INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal - 500 043, Hyderabad, Telangana

COURSE CONTENT

DATA STRUCTURES LABORATORY								
III Semester: AE / ME / CE / ECE / EEE / CSE / CSE (AI & ML) / CSE (DS) / CSE (CS) / IT								
Course Code	Category	Hours / Week			Credits	Maximum Marks		
ACSD11	Core	L	T	P	C	CIA	SEE	Total
		-	-	2	1	40	60	100
Contact Classes: Nil	Tutorial Classes: Nil	Practical Classes: 45			Total Classes: 45			
Prerequisite: Essentials of Problem Solving								

I. COURSE OVERVIEW:

The course covers some of the general-purpose data structures and algorithms, and software development. Topics covered include managing complexity, analysis, static data structures, dynamic data structures and hashing mechanisms. The main objective of the course is to teach the students how to select and design data structures and algorithms that are appropriate for problems that they might encounter in real life. This course reaches to student by power point presentations, lecture notes, and lab which involve the problem solving in mathematical and engineering areas.

II. COURSE OBJECTIVES:

The students will try to learn

- To provide students with skills needed to understand and analyze performance trade-offs of different algorithms / implementations and asymptotic analysis of their running time and memory usage.
- To provide knowledge of basic abstract data types (ADT) and associated algorithms: stacks, queues, lists, tree, graphs, hashing and sorting, selection and searching.
- The fundamentals of how to store, retrieve, and process data efficiently.
- To provide practice by specifying and implementing these data structures and algorithms in Python.
- Understand essential for future programming and software engineering courses.

III. COURSE OUTCOMES:

At the end of the course students should be able to:

- CO 1 Interpret the complexity of algorithm using the asymptotic notations.
- CO 2 Select appropriate searching and sorting technique for a given problem.
- CO 3 Construct programs on performing operations on linear and nonlinear data structures for organization of data.
- CO 4 Make use of linear data structures and nonlinear data structures solving real time applications.
- CO 5 Describe hashing techniques and collision resolution methods for efficiently accessing data with respect to performance.
- CO 6 Compare various types of data structures; in terms of implementation, operations and performance.

DATA STRUCTURES LABORATORY COURSE CONTENT

S No.	Topic Name
1.	Getting Started Exercises <ul style="list-style-type: none">a. Sum of last digits of two given numbersb. Is N an exact multiple of M?c. Combine Stringsd. Even or Odde. Second last digit of a given numberf. Alternate String Combinerg. Padovan Sequenceh. Leaders in an arrayi. Find the Value of a Number Raised to its Reversej. Mean of Array using Recursion
2.	Searching <ul style="list-style-type: none">a. Linear / Sequential Searchb. Binary Searchc. Uniform Binary Searchd. Interpolation Searche. Fibonacci Search
3.	Sorting <ul style="list-style-type: none">a. Bubble Sortingb. Selection Sortc. Insertion Sort
4.	Divide and Conquer <ul style="list-style-type: none">a. Quick Sortb. Merge Sortc. Heap Sortd. Radix Sorte. Shell Sort
5.	Stack <ul style="list-style-type: none">a. Implementation of Stackb. Balanced Parenthesis Checkingc. Evaluation of Postfix Expressiond. Infix to Postfix Expression Conversione. Reverse a Stack
6.	Queue <ul style="list-style-type: none">a. Linear Queueb. Stack using Queuesc. Queue using Stacksd. Circular Queuee. Deque (Doubly Ended Queue)
7.	Linked List <ul style="list-style-type: none">a. Singly Linked Listb. Linked List Cyclec. Remove Linked List Elementsd. Reverse Linked Liste. Palindrome Linked Listf. Middle of the Linked Listg. Convert Binary Number in a Linked List to Integer

- 8. Circular Single Linked List and Doubly Linked List**
 - a. Circular Linked List
 - b. Doubly Linked List
 - c. Sorted Merge of Two Sorted Doubly Circular Linked Lists
 - d. Delete all occurrences of a given key in a Doubly Linked List
 - e. Delete a Doubly Linked List Node at a Given Position
- 9. Trees**
 - a. Tree Creation and Basic Tree Terminologies
 - b. Binary Tree Traversal Techniques
 - c. Insertion in a Binary Tree in Level Order
 - d. Finding the Maximum Height or Depth of a Binary Tree
 - e. Deletion in a Binary Tree
- 10. Binary Search Tree (BST)**
 - a. Searching in Binary Search Tree
 - b. Find the node with Minimum Value in a BST
 - c. Check if a Binary Tree is BST or not
 - d. Second Largest Element in BST
 - e. Insertion in Binary Search Tree (BST)
- 11. AVL Tree**
 - a. Insertion in an AVL Tree
 - b. Deletion in an AVL Tree
 - c. Count Greater Nodes in AVL Tree
 - d. Minimum Number of Nodes in an AVL Tree with given Height
- 12. Graph Traversal**
 - a. Breadth First Search
 - b. Depth First Search
 - c. Best First Search (Informed Search)
 - d. Breadth First Traversal of a Graph
 - e. Depth First Search (DFS) for Disconnected Graph
- 13. Minimum Spanning Tree (MST)**
 - a. Kruskal's Algorithm
 - b. Prim's Algorithm
 - c. Total Number of Spanning Trees in a Graph
 - d. Minimum Product Spanning Tree
- 14. Final Notes**

IV. COURSE CONTENT:

EXERCISES FOR DATA STRUCTURES LABORATORY

Note: Students are encouraged to bring their own laptops for laboratory practice sessions.

1. Getting Started Exercises

1.1 Sum of last digits of two given numbers

Rohit wants to add the last digits of two given numbers. For example, If the given numbers are 267 and 154, the output should be 11.

Below is the explanation -

Last digit of the 267 is 7

Last digit of the 154 is 4

Sum of 7 and 4 = 11

Write a program to help Rohit achieve this for any given two numbers.

The prototype of the method should be -

int addLastDigits(int input1, int input2);

where input1 and input2 denote the two numbers whose last digits are to be added.

Note: The sign of the input numbers should be ignored.

if the input numbers are 267 and 154, the sum of last two digits should be 11

if the input numbers are 267 and -154, the sum of last two digits should be 11

if the input numbers are -267 and 154, the sum of last two digits should be 11

if the input numbers are -267 and -154, the sum of last two digits should be 11

Input: 267 154

Output: 11

Input: 267 -154

Output: 11

Input: -267 154

Output: 11

Input: -267 -154

Output: 11

```
import java.util.Scanner;

class AddLastDigitsFunction
{
    int addLastDigits(int n1, int n2)
    {
        # Write code here
    }

    public static void main(String args[])
    {
        AddLastDigitsFunction obj = new AddLastDigitsFunction();
        # Write code here
        System.out.println(obj.addLastDigits(n1,n2));
    }
}
```

1.2 Is N an exact multiple of M?

Write a function that accepts two parameters and finds whether the first parameter is an exact multiple of the second parameter. If the first parameter is an exact multiple of the second parameter, the function should return 2 else it should return 1.

If either of the parameters are zero, the function should return 3.

Assumption: Within the scope of this question, assume that - the first parameter can be positive, negative or zero the second parameter will always be ≥ 0

Input: num1 = 10, num2 = 5

Output: 2

Input: num1 = -10, num2 = 5

Output: 2

Input: num1 = 0, num2 = 5

Output: 3

Input: num1 = 10, num2 = 3

Output: 1

```
public class MultipleChecker
{
    public static int checkMultiple(int num1, int num2)
    {
        # Write code here
    }

    public static void main(String[] args)
    {
        # Write code here
    }
}
```

1.3 Combine Strings

Given 2 strings, a and b, return a new string of the form short+long+short, with the shorter string on the outside and the longer string in the inside. The strings will not be the same length, but they may be empty (length 0).

If input is "hi" and "hello", then output will be "hihellohi"

Input: Enter the first string: "hi"

Enter the second string: "hello"

Output: "hihellohi"

Input: Enter the first string: "iare"

Enter the second string: "college"

Output: "iarecollegeiare"

```
public class StringCombiner
{
    public static void main(String[] args)
    {
        # Write code here
    }

    public static String combineStrings(String a, String b)
    {
        # Write code here
    }
}
```

1.4 Even or Odd

Write a function that accepts 6 input parameters. The first 5 input parameters are of type int. The sixth input parameter is of type string. If the sixth parameter contains the value "even", the function is supposed to return the count of how many of the first five input parameters are even. If the sixth parameter contains the value "odd", the function is supposed to return the count of how many of the first five input parameters are odd.

Example:

If the five input parameters are 12, 17, 19, 14, and 115, and the sixth parameter is "odd", the function must return 3, because there are three odd numbers 17, 19 and 115.

If the five input parameters are 12, 17, 19, 14, and 115, and the sixth parameter is "even", the function must return 2, because there are two even numbers 12 and 14.

Note that zero is considered an even number.

Input: num1 = 12;
num2 = 17;
num3 = 19;
num4 = 14;
num5 = 115;
type = "odd"

Output: 3

Input: num1 = 12;
num2 = 17;
num3 = 19;
num4 = 14;
num5 = 115;
type = "even"

Output: 2

```
public class NumberCounter
{
    public static int countNumbers(int num1, int num2, int num3, int num4, int num5,
String type)
    {
        # Write code here
    }

    public static void main(String[] args)
    {
        # Write code here
    }
}
```

1.5 Second last digit of a given number

Write a function that returns the second last digit of the given number. Second last digit is being referred to the digit in the tens place in the given number.

Example: if the given number is 197, the second last digit is 9.

Note 1: The second last digit should be returned as a positive number. i.e. if the given number is -197, the second last digit is 9.

Note 2: If the given number is a single digit number, then the second last digit does not exist. In such cases, the function should return -1. i.e. if the given number is 5, the second last digit should be returned as -1.

Input: 197

Output: 9

Input: 5

Output: -1

Input: -197

Output: 9

```
public class SecondLastDigit
{
    public static int getSecondLastDigit(int number)
    {
        # write code here
    }
    public static void main(String[] args)
    {
        # write code here
    }
}
```

1.6 Alternate String Combiner

Given two strings, a and b, print a new string which is made of the following combination-first character of a, the first character of b, second character of a, second character of b and so on.

Any characters left, will go to the end of the result.

Hello,World

HWeolrllod

Input: "Hello,World"

Output: "HWeolrllod"

Input: "Iare,College"

Output: "lCaorlelege"

```
public class AlternateStringCombiner
{
    public static void main(String[] args)
    {
        # write code here
    }
    public static String combineStrings(String a, String b)
    {
        # write code here
    }
}
```

1.7 Padovan Sequence

The Padovan sequence is a sequence of numbers named after Richard Padovan, who attributed its discovery to Dutch architect Hans van der Laan. The sequence was described by Ian Stewart in his Scientific American column Mathematical Recreations in June 1996. The Padovan sequence is defined by the following recurrence relation:

$$P(n) = P(n-2) + P(n-3)$$

with the initial conditions $P(0) = P(1) = P(2) = 1$.

In this sequence, each term is the sum of the two preceding terms, similar to the Fibonacci sequence.

However, the Padovan sequence has different initial conditions and exhibits different growth patterns.

The first few terms of the Padovan sequence are: 1, 1, 1, 2, 2, 3, 4, 5, 7, 9, 12, ...

Input: num = 10

Output: Padovan Sequence up to 10:

1 1 1 2 2 3 4 5 7 9 12

Input: num = 20

Output: Padovan Sequence up to 20:

1 1 1 2 2 3 4 5 7 9 12 16 21 28 37 49 65 86 114 151 200

```
public class PadovanSequence
{
    public static int padovan(int n)
    {
        # write code here
    }
    public static void main(String[] args)
    {
        # write code here
    }
}
```

1.8 Leaders in an array

Given an array arr of n positive integers, your task is to find all the leaders in the array. An element of the array is considered a leader if it is greater than all the elements on its right side or if it is equal to the maximum element on its right side. The rightmost element is always a leader.

Input: n = 6, arr[] = {16, 17, 4, 3, 5, 2}

Output: 17 5 2

Input: n = 5, arr[] = {10, 4, 2, 4, 1}

Output: 10 4 4 1

Input: n = 4, arr[] = {5, 10, 20, 40}

Output: 40

Input: n = 4, arr[] = {30, 10, 10, 5}

Output: 30 10 10 5

```
import java.util.ArrayList;
import java.util.List;
public class ArrayLeaders
{
    public static List<Integer> findArrayLeaders(int[] arr)
    {
        # write code here
    }
    public static void main(String[] args)
    {
        # write code here
    }
}
```

1.9 Find the Value of a Number Raised to its Reverse

Given a number N and its reverse R. The task is to find the number obtained when the number is raised to the power of its own reverse

Input : N = 2, R = 2

Output: 4

Explanation: Number 2 raised to the power of its reverse 2 gives 4 which gives 4 as a result after performing modulo 10^9+7

Input: N = 57, R = 75

Output: 262042770

Explanation: 57^{75} modulo 10^9+7 gives us the result as 262042770

```
public class NumberPower
{
    public static long powerOfReverse(int N, int R)
```

```

{
    # write code here
}
public static void main(String[] args)
{
    # write code here
}
}

```

1.10 Mean of Array using Recursion

Find the mean of the elements of the array.

Mean = (Sum of elements of the Array) / (Total no of elements in Array)

Input: 1 2 3 4 5

Output: 3.0

Input: 1 2 3

Output: 2.0

To find the mean using recursion assume that the problem is already solved for N-1 i.e. you have to find for n

Sum of first N-1 elements = (Mean of N-1 elements) * (N-1)

Mean of N elements = (Sum of first N-1 elements + N-th elements) / (N)

```

public class ArrayMean
{
    public static double findArrayMean(int[] arr)
    {
        # write code here
    }
    public static void main(String[] args)
    {
        # write code here
    }
}

```

Try:

1. **Kth Smallest Element:** Given an array arr[] and an integer k where k is smaller than the size of the array, the task is to find the kth smallest element in the given array. It is given that all array elements are distinct.

Note: l and r denotes the starting and ending index of the array.

Input: n = 6, arr[] = {7, 10, 4, 3, 20, 15}, k = 3, l = 0, r = 5

Output: 7

Explanation: 3rd smallest element in the given array is 7.

Input: n = 5, arr[] = {7, 10, 4, 20, 15}, k = 4, l=0 r=4

Output: 15

Explanation: 4th smallest element in the given array is 15.

Your task is to complete the function **kthSmallest()** which takes the array arr[], integers l and r denoting the starting and ending index of the array and an integer k as input and returns the kth smallest element.

2. **Count pairs with given sum:** Given an array of N integers, and an integer K, find the number of pairs of elements in the array whose sum is equal to K. Your task is to complete the function **getPairsCount()** which takes arr[], n and k as input parameters and returns the number of pairs that have sum K.

Input: N = 4, K = 6, arr[] = {1, 5, 7, 1}

Output: 2

Explanation: $\text{arr}[0] + \text{arr}[1] = 1 + 5 = 6$ and $\text{arr}[1] + \text{arr}[3] = 5 + 1 = 6$.

Input: $N = 4, K = 2, \text{arr}[] = \{1, 1, 1, 1\}$

Output: 6

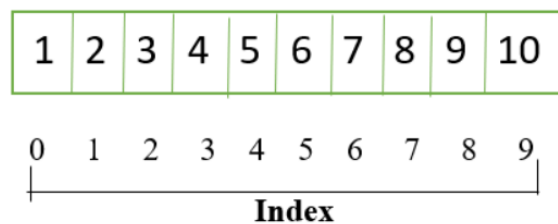
Explanation: Each 1 will produce sum 2 with any 1.

2. Searching

2.1 Linear / Sequential Search

Linear search is defined as the searching algorithm where the list or data set is traversed from one end to find the desired value. Given an array $\text{arr}[]$ of n elements, write a recursive function to search a given element x in $\text{arr}[]$.

Find '6'



1	2	3	4	5	6	7	8	9	10
0	1	2	3	4	5	6	7	8	9
Index									

Note : We find '6' at index '5' through linear search

Linear search procedure:

1. Start from the leftmost element of $\text{arr}[]$ and one by one compare x with each element of $\text{arr}[]$
2. If x matches with an element, return the index.
3. If x doesn't match with any of the elements, return -1.

Input: $\text{arr}[] = \{10, 20, 80, 30, 60, 50, 110, 100, 130, 170\}$

$x = 110;$

Output: 6

Element x is present at index 6

Input: $\text{arr}[] = \{10, 20, 80, 30, 60, 50, 110, 100, 130, 170\}$

$x = 175;$

Output: -1

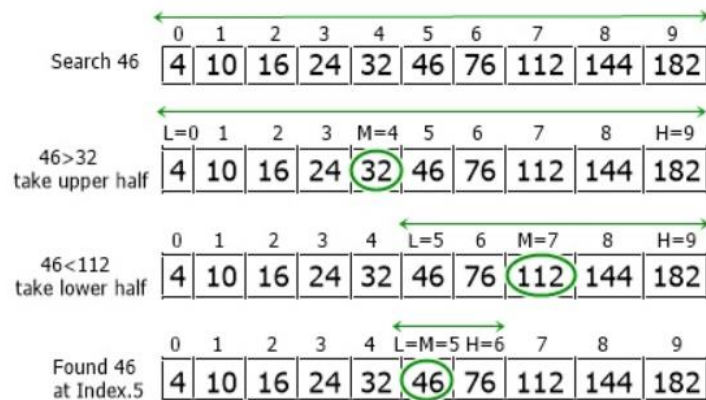
Element x is not present in $\text{arr}[]$.

```
public class RecursiveLinearSearch
{
    public static int recursiveLinearSearch(int[] arr, int key, int index)
    {
        # write code here
    }

    public static void main(String[] args)
    {
        # write code here
    }
}
```

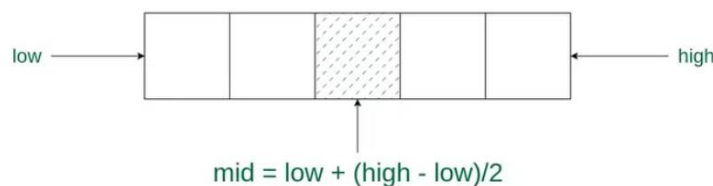
2.2 Binary Search

Binary Search is defined as a searching algorithm used in a sorted array by repeatedly dividing the search interval in half. The idea of binary search is to use the information that the array is sorted and reduce the time complexity to $O(\log N)$.



Conditions for Binary Search algorithm:

1. The data structure must be sorted.
2. Access to any element of the data structure takes constant time.



Binary Search Procedure:

1. Divide the search space into two halves by finding the middle index "mid".
2. Compare the middle element of the search space with the key.
3. If the key is found at middle element, the process is terminated.
4. If the key is not found at middle element, choose which half will be used as the next search space.
 - a. If the key is smaller than the middle element, then the left side is used for next search.
 - b. If the key is larger than the middle element, then the right side is used for next search.
5. This process is continued until the key is found or the total search space is exhausted.

Input: arr = [2, 5, 8, 12, 16, 23, 38, 56, 72, 91]

Output: target = 23

Element 23 is present at index 5

```
public class RecursiveBinarySearch
{
    public static int recursiveBinarySearch(int[] arr, int key, int left, int right)
    {
        # write code here
    }

    public static void main(String[] args)
    {
        # write code here
    }
}
```


2.3 Uniform Binary Search

Uniform Binary Search is an optimization of Binary Search algorithm when many searches are made on same array or many arrays of same size. In normal binary search, we do arithmetic operations to find the mid points. Here we precompute mid points and fills them in lookup table. The array look-up generally works faster than arithmetic done (addition and shift) to find the mid-point.

Input: array = {1, 3, 5, 6, 7, 8, 9}, v=3

Output: Position of 3 in array = 2

Input: array = {1, 3, 5, 6, 7, 8, 9}, v=7

Output: Position of 7 in array = 5

The algorithm is very similar to Binary Search algorithm, the only difference is a lookup table is created for an array and the lookup table is used to modify the index of the pointer in the array which makes the search faster. Instead of maintaining lower and upper bound the algorithm maintains an index and the index is modified using the lookup table.

```
public class RecursiveUniformBinarySearch
{
    public static int recursiveUniformBinarySearch(int[] arr, int key, int[]
lookupTable, int left, int right)
    {
        # write code here
    }
    public static void main(String[] args)
    {
        # write code here
    }
}
```

2.4 Interpolation Search

Interpolation search works better than Binary Search for a Sorted and Uniformly Distributed array. Binary search goes to the middle element to check irrespective of search-key. On the other hand, Interpolation search may go to different locations according to search-key. If the value of the search-key is close to the last element, Interpolation Search is likely to start search toward the end side. Interpolation search is more efficient than binary search when the elements in the list are uniformly distributed, while binary search is more efficient when the elements in the list are not uniformly distributed.

Interpolation search can take longer to implement than binary search, as it requires the use of additional calculations to estimate the position of the target element.

Input: arr = [1, 2, 3, 4, 5, 6, 7, 8, 9]

Output: target = 5

```
public class InterpolationSearch
{
    public static int interpolationSearch(int[] arr, int key)
    {
        # write code here
    }

    public static void main(String[] args)
    {
```

```
}  
    # write code here  
}
```

2.5 Fibonacci Search

Given a sorted array `arr[]` of size `n` and an element `x` to be searched in it. Return index of `x` if it is present in array else return `-1`.

Input: `arr[] = {2, 3, 4, 10, 40}`, `x = 10`

Output: 3

Element `x` is present at index 3.

Input: `arr[] = {2, 3, 4, 10, 40}`, `x = 11`

Output: -1

Element `x` is not present.

Fibonacci Search is a comparison-based technique that uses Fibonacci numbers to search an element in a sorted array.

Fibonacci Numbers are recursively defined as $F(n) = F(n-1) + F(n-2)$, $F(0) = 0$, $F(1) = 1$. First few Fibonacci Numbers are 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ...

Fibonacci Search Procedure:

Let the searched element be `x`. The idea is to first find the smallest Fibonacci number that is greater than or equal to the length of the given array. Let the found Fibonacci number be `fib` (`m`'th Fibonacci number). We use (`m-2`)'th Fibonacci number as the index (If it is a valid index). Let (`m-2`)'th Fibonacci Number be `i`, we compare `arr[i]` with `x`, if `x` is same, we return `i`. Else if `x` is greater, we recur for subarray after `i`, else we recur for subarray before `i`.

Let `arr[0..n-1]` be the input array and the element to be searched be `x`.

1. Find the smallest Fibonacci number greater than or equal to `n`. Let this number be `fibM` [`m`'th Fibonacci number]. Let the two Fibonacci numbers preceding it be `fibMm1` [(`m-1`)'th Fibonacci Number] and `fibMm2` [(`m-2`)'th Fibonacci Number].
2. While the array has elements to be inspected:
 - i. Compare `x` with the last element of the range covered by `fibMm2`
 - ii. If `x` matches, return index
 - iii. Else If `x` is less than the element, move the three Fibonacci variables two Fibonacci down, indicating elimination of approximately rear two-third of the remaining array.
 - iv. Else `x` is greater than the element, move the three Fibonacci variables one Fibonacci down. Reset offset to index. Together these indicate the elimination of approximately front one-third of the remaining array.
3. Since there might be a single element remaining for comparison, check if `fibMm1` is 1. If Yes, compare `x` with that remaining element. If match, return index.

```
public class FibonacciSearch  
{  
    public static int fibonacciSearch(int[] arr, int key)  
    {  
        # write code here  
    }  
  
    public static void main(String[] args)  
    {  
        # write code here  
    }  
}
```

```
}  
}
```

3. Sorting

3.1 Bubble Sort

Bubble Sort is the simplest sorting algorithm that works by repeatedly swapping the adjacent elements if they are in the wrong order. This algorithm is not suitable for large data sets as its average and worst-case time complexity is quite high.

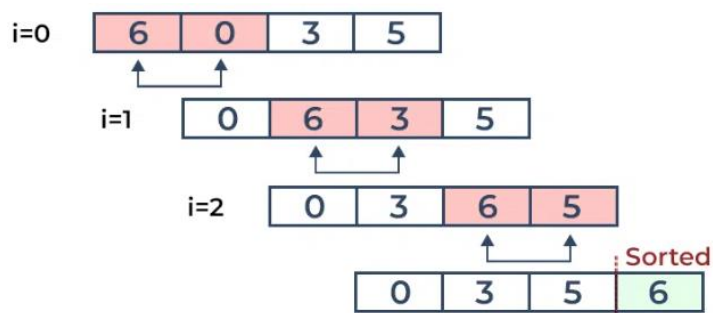
Bubble Sort Procedure:

1. Traverse from left and compare adjacent elements and the higher one is placed at right side.
2. In this way, the largest element is moved to the rightmost end at first.
3. This process is then continued to find the second largest and place it and so on until the data is sorted.

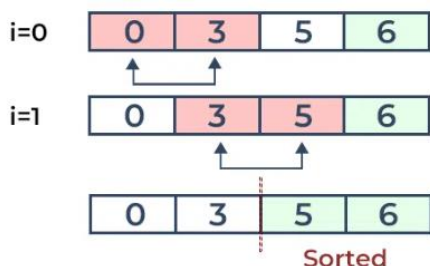
Input: arr = [6, 3, 0, 5]

Output:

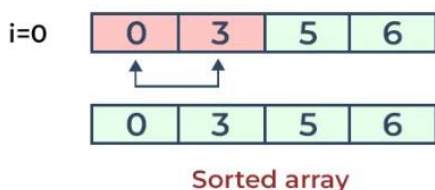
First Pass:



Second Pass:



Third Pass:



```
import java.util.Scanner;
```

```

class BubbleSortExample
{
    public static void main(String[] args)
    {
        # write code here

    }

    public static void bubbleSort(int[] arr)
    {
        # write code here

    }
}

```

3.2 Selection Sort

Selection sort is a simple and efficient sorting algorithm that works by repeatedly selecting the smallest (or largest) element from the unsorted portion of the list and moving it to the sorted portion of the list. The algorithm repeatedly selects the smallest (or largest) element from the unsorted portion of the list and swaps it with the first element of the unsorted part. This process is repeated for the remaining unsorted portion until the entire list is sorted.

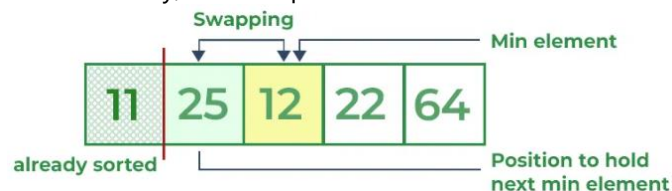
Input: arr = [64, 25, 12, 22, 11]

Output: arr = [11, 12, 22, 25, 64]

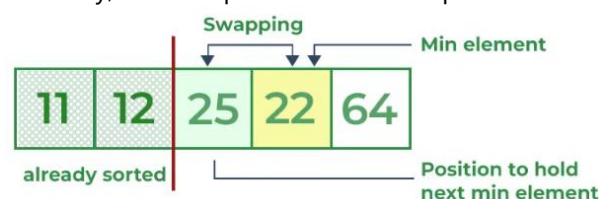
First Pass: For the first position in the sorted array, the whole array is traversed from index 0 to 4 sequentially. The first position where 64 is stored presently, after traversing whole array it is clear that 11 is the lowest value. Thus, replace 64 with 11. After one iteration 11, which happens to be the least value in the array, tends to appear in the first position of the sorted list.



Second Pass: For the second position, where 25 is present, again traverse the rest of the array in a sequential manner. After traversing, we found that 12 is the second lowest value in the array and it should appear at the second place in the array, thus swap these values.



Third Pass: Now, for third place, where 25 is present again traverse the rest of the array and find the third least value present in the array. While traversing, 22 came out to be the third least value and it should appear at the third place in the array, thus swap 22 with element present at third position.



Fourth Pass: Similarly, for fourth position traverse the rest of the array and find the fourth least element in the array. As 25 is the 4th lowest value hence, it will place at the fourth position.



Fifth Pass: At last the largest value present in the array automatically get placed at the last position in the array. The resulted array is the sorted array.



```
import java.util.Scanner;

class SelectionSortExample
{
    public static void main(String[] args)
    {
        # write code here
    }

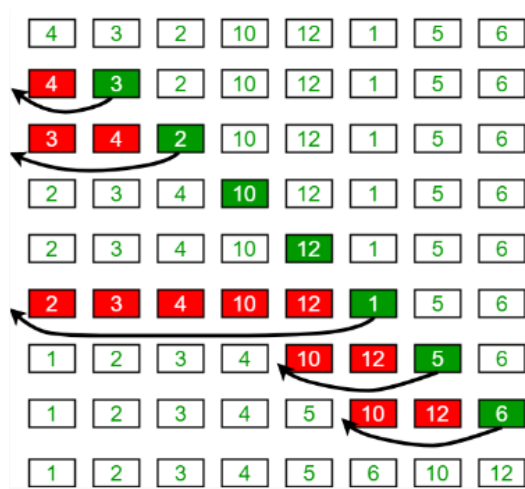
    public static void selectionSort(int[] arr)
    {
        # write code here
    }
}
```

3.3 Insertion Sort

Insertion sort is a simple sorting algorithm that works similar to the way you sort playing cards in your hands. The array is virtually split into a sorted and an unsorted part. Values from the unsorted part are picked and placed at the correct position in the sorted part.

Insertion Sort Procedure:

1. To sort an array of size N in ascending order iterate over the array and compare the current element (key) to its predecessor, if the key element is smaller than its predecessor, compare it to the elements before.
2. Move the greater elements one position up to make space for the swapped element.



Input: arr = [4, 3, 2, 10, 12, 1, 5, 6]

Output: arr = [1, 2, 3, 4, 5, 6, 10, 12]

```
import java.util.Scanner;

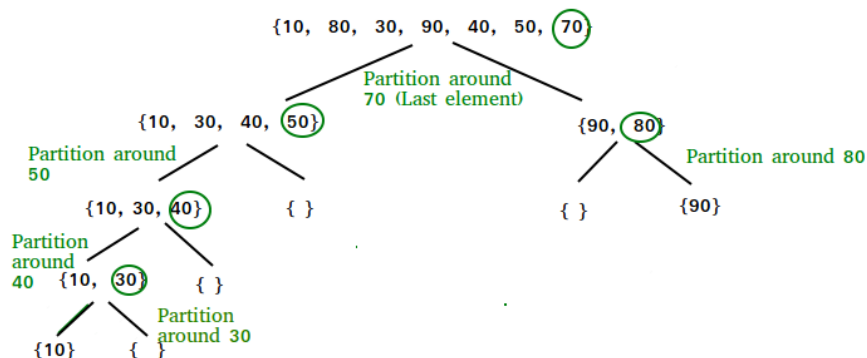
class InsertionSortExample
{
    public static void main(String[] args)
    {
        # write code here
    }

    public static void insertionSort(int[] arr)
    {
        # write code here
    }
}
```

4. Divide and Conquer

4.1 Quick Sort

QuickSort is a sorting algorithm based on the Divide and Conquer algorithm that picks an element as a pivot and partitions the given array around the picked pivot by placing the pivot in its correct position in the sorted array. The key process in quickSort is a partition(). The target of partitions is to place the pivot (any element can be chosen to be a pivot) at its correct position in the sorted array and put all smaller elements to the left of the pivot, and all greater elements to the right of the pivot. Partition is done recursively on each side of the pivot after the pivot is placed in its correct position and this finally sorts the array.



The quick sort method can be summarized in three steps:

1. **Pick:** Select a pivot element.
2. **Divide:** Split the problem set, move smaller parts to the left of the pivot and larger items to the right.
3. **Repeat and combine:** Repeat the steps and combine the arrays that have previously been sorted.

Algorithm for Quick Sort Function:

```
//start --> Starting index, end --> Ending index
Quicksort(array, start, end)
{
    if (start < end)
    {
        pIndex = Partition(A, start, end)
        Quicksort(A, start, pIndex-1)
        Quicksort(A, pIndex+1, end)
    }
}
```

Algorithm for Partition Function:

```
partition (array, start, end)
{
    // Setting rightmost Index as pivot
    pivot = arr[end];

    i = (start - 1) // Index of smaller element and indicates the
    // right position of pivot found so far
    for (j = start; j <= end- 1; j++)
    {
```

```

        // If current element is smaller than the pivot
        if (arr[j] < pivot)
        {
            i++; // increment index of smaller element
            swap arr[i] and arr[j]
        }
    }
    swap arr[i + 1] and arr[end])
    return (i + 1)
}

```

Input: arr = [10, 80, 30, 90, 40, 50, 70]

Output: arr = [10, 30, 40, 50, 70, 80, 90]

```

import java.util.Scanner;

class QuickSortExample
{
    public static void main(String[] args)
    {
        # write code here
    }

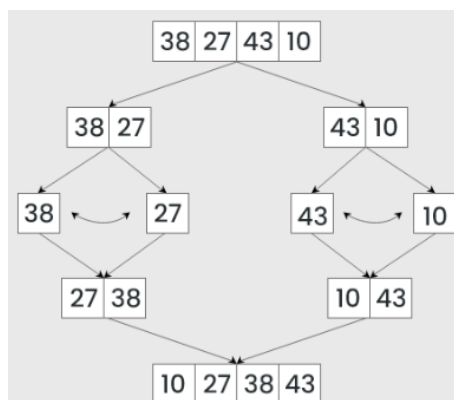
    public static void quickSort(int[] arr, int low, int high)
    {
        # write code here
    }

    public static int partition(int[] arr, int low, int high)
    {
        # write code here
    }
}

```

4.2 Merge Sort

Merge sort is defined as a sorting algorithm that works by dividing an array into smaller subarrays, sorting each subarray, and then merging the sorted subarrays back together to form the final sorted array. In simple terms, we can say that the process of merge sort is to divide the array into two halves, sort each half, and then merge the sorted halves back together. This process is repeated until the entire array is sorted.



Input: arr = [12, 11, 13, 5, 6, 7]

Output: arr = [5, 6, 7, 11, 12, 13]

```
import java.util.Scanner;

class MergeSortExample
{
    public static void main(String[] args)
    {
        # write code here
    }

    public static void mergeSort(int[] arr, int low, int high)
    {
        # write code here
    }

    public static void merge(int[] arr, int low, int mid, int high)
    {
        # write code here
    }
}
```

4.3 Heap Sort

Heap sort is a comparison-based sorting technique based on Binary Heap data structure. It is similar to the selection sort where we first find the minimum element and place the minimum element at the beginning. Repeat the same process for the remaining elements.

Heap Sort Procedure:

First convert the array into heap data structure using heapify, then one by one delete the root node of the Max-heap and replace it with the last node in the heap and then heapify the root of the heap. Repeat this process until size of heap is greater than 1.

- Build a heap from the given input array.
- Repeat the following steps until the heap contains only one element:
 - Swap the root element of the heap (which is the largest element) with the last element of the heap.
 - Remove the last element of the heap (which is now in the correct position).
 - Heapify the remaining elements of the heap.
- The sorted array is obtained by reversing the order of the elements in the input array.

Input: arr = [12, 11, 13, 5, 6, 7]

Output: Sorted array is 5 6 7 11 12 13

```
import java.util.Scanner;

class HeapSortExample
{
    public static void main(String[] args)
    {
        # write code here
    }

    public static void heapSort(int[] arr)
    {
```

```

        # write code here
    }

    public static void heapify(int[] arr, int n, int i)
    {
        # write code here
    }
}

```

4.4 Radix Sort

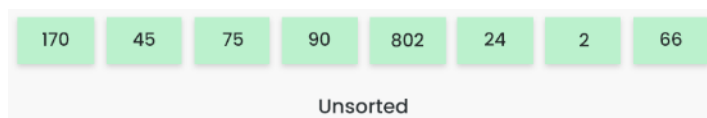
Radix Sort is a linear sorting algorithm that sorts elements by processing them digit by digit. It is an efficient sorting algorithm for integers or strings with fixed-size keys. Rather than comparing elements directly, Radix Sort distributes the elements into buckets based on each digit's value. By repeatedly sorting the elements by their significant digits, from the least significant to the most significant, Radix Sort achieves the final sorted order.

Radix Sort Procedure:

The key idea behind Radix Sort is to exploit the concept of place value.

1. It assumes that sorting numbers digit by digit will eventually result in a fully sorted list.
2. Radix Sort can be performed using different variations, such as Least Significant Digit (LSD) Radix Sort or Most Significant Digit (MSD) Radix Sort.

To perform radix sort on the array [170, 45, 75, 90, 802, 24, 2, 66], we follow these steps:



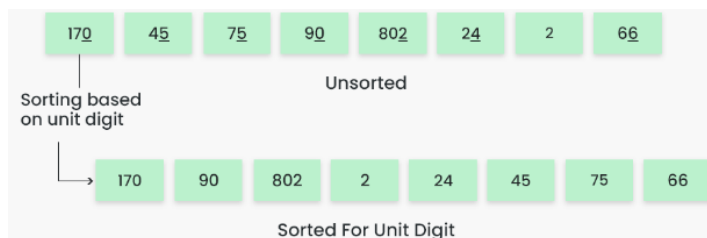
Step 1: Find the largest element in the array, which is 802. It has three digits, so we will iterate three times, once for each significant place.

Step 2: Sort the elements based on the unit place digits ($X=0$). We use a stable sorting technique, such as counting sort, to sort the digits at each significant place.

Sorting based on the unit place:

Perform counting sort on the array based on the unit place digits.

The sorted array based on the unit place is [170, 90, 802, 2, 24, 45, 75, 66]

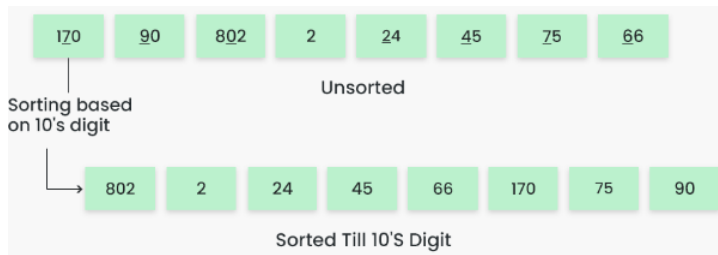


Step 3: Sort the elements based on the tens place digits.

Sorting based on the tens place:

Perform counting sort on the array based on the tens place digits.

The sorted array based on the tens place is [802, 2, 24, 45, 66, 170, 75, 90]

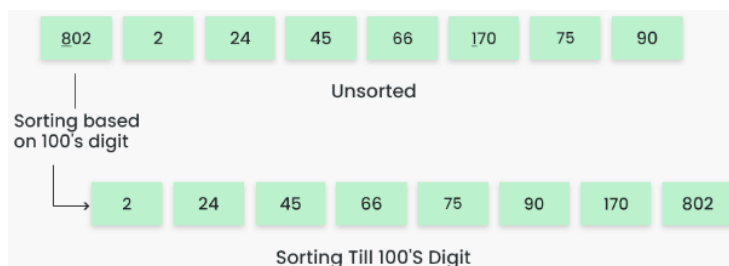


Step 4: Sort the elements based on the hundreds place digits.

Sorting based on the hundreds place:

Perform counting sort on the array based on the hundreds place digits.

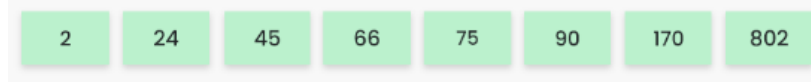
The sorted array based on the hundreds place is [2, 24, 45, 66, 75, 90, 170, 802]



Step 5: The array is now sorted in ascending order.

The final sorted array using radix sort is [2, 24, 45, 66, 75, 90, 170, 802]

Array after performing **Radix Sort** for all digits



```
import java.util.Arrays;

class RadixSortExample
{
    public static void main(String[] args)
    {
        # write code here
    }

    public static void radixSort(int[] arr)
    {
        # write code here
    }

    public static int getMax(int[] arr)
    {
        # write code here
    }

    public static void countSort(int[] arr, int exp)
    {
        # write code here
    }
}
```

4.5 Shell Sort

Shell sort is mainly a variation of Insertion Sort. In insertion sort, we move elements only one position ahead. When an element has to be moved far ahead, many movements are involved. The idea of ShellSort is to allow the exchange of far items. In Shell sort, we make the array h-sorted for a large value of h. We keep reducing the value of h until it becomes 1. An array is said to be h-sorted if all sublists of every h'th element are sorted.

Shell Sort Procedure:

1. Initialize the value of gap size h
2. Divide the list into smaller sub-part. Each must have equal intervals to h
3. Sort these sub-lists using insertion sort
4. Repeat this step 1 until the list is sorted.
5. Print a sorted list.

```
Procedure Shell_Sort(Array, N)
  While Gap < Length(Array) /3 :
    Gap = ( Interval * 3 ) + 1
  End While Loop
  While Gap > 0 :
    For (Outer = Gap; Outer < Length(Array); Outer++):
      Insertion_Value = Array[Outer]
      Inner = Outer;
      While Inner > Gap-1 And Array[Inner - Gap] >= Insertion_Value:
        Array[Inner] = Array[Inner - Gap]
        Inner = Inner - Gap
      End While Loop
      Array[Inner] = Insertion_Value
    End For Loop
    Gap = (Gap -1) /3;
  End While Loop
End Shell_Sort
```

```
import java.util.Scanner;

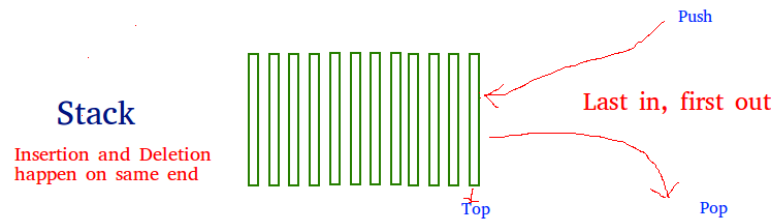
class ShellSortExample
{
    public static void main(String[] args)
    {
        # write code here
    }

    public static void shellSort(int[] arr)
    {
        # write code here
    }
}
```

5. Stack

5.1 Implementation of Stack

A stack is a linear data structure that stores items in a Last-In/First-Out (LIFO) or First-In/Last-Out (FILO) manner. In stack, a new element is added at one end and an element is removed from that end only. The insert and delete operations are often called push and pop.



The functions associated with stack are:

- **empty()** – Returns whether the stack is empty
- **size()** – Returns the size of the stack
- **top() / peek()** – Returns a reference to the topmost element of the stack
- **push(a)** – Inserts the element 'a' at the top of the stack
- **pop()** – Deletes the topmost element of the stack

```
class Stack
{
    private int maxSize;
    private int top;
    private int[] stackArray;

    public Stack(int size)
    {
        # write code here
    }

    public void push(int value)
    {
        # write code here
    }

    public int pop()
    {
        # write code here
    }

    public int peek()
    {
        # write code here
    }

    public boolean isEmpty()
    {
        # write code here
    }

    public boolean isFull()
    {
        # write code here
    }
}
```

```
class StackExample
{
```

```

public static void main(String[] args)
{
    Stack stack = new Stack(5);
    stack.push(10);
    stack.push(20);
    stack.push(30);
    stack.pop();
    stack.peek();
    stack.push(40);
    stack.push(50);
    stack.push(60);
}
}

```

5.2 Balanced Parenthesis Checking

Given an expression string, write a java program to find whether a given string has balanced parentheses or not.

Input: "{(a+b)*(c-d)}"

Output: true

Input: "{(a+b)*[c-d]}"

Output: false

One approach to check balanced parentheses is to use stack. Each time, when an open parentheses is encountered push it in the stack, and when closed parenthesis is encountered, match it with the top of stack and pop it. If stack is empty at the end, return true otherwise, false

```

import java.util.Stack;

class BalancedParenthesisChecker
{
    public static boolean isBalanced(String expression)
    {
        # write code here
    }

    public static void main(String[] args)
    {
        String expression1 = "{(a+b)*(c-d)}";
        String expression2 = "{(a+b)*[c-d]}";

        # write code here
    }
}

```

5.3 Evaluation of Postfix Expression

Given a postfix expression, the task is to evaluate the postfix expression. Postfix expression: The expression of the form "a b operator" (ab+) i.e., when a pair of operands is followed by an operator.

Input: str = "2 3 1 * + 9 -"

Output: -4

Explanation: If the expression is converted into an infix expression, it will be $2 + (3 * 1) - 9 = 5 - 9 = -4$.

Input: str = "100 200 + 2 / 5 * 7 +"

Output: 757

Procedure for evaluation postfix expression using stack:

- Create a stack to store operands (or values).
- Scan the given expression from left to right and do the following for every scanned element.
 - If the element is a number, push it into the stack.

- If the element is an operator, pop operands for the operator from the stack. Evaluate the operator and push the result back to the stack.
- When the expression is ended, the number in the stack is the final answer.

```
import java.util.Stack;

class PostfixEvaluator
{
    public static int evaluatePostfix(String expression)
    {
        # write code here
    }

    public static int performOperation(char operator, int operand1, int operand2)
    {
        # write code here
    }

    public static void main(String[] args)
    {
        # write code here
    }
}
```

5.4 Infix to Postfix Expression Conversion

For a given Infix expression, convert it into Postfix form.

Infix expression: The expression of the form "a operator b" (a + b) i.e., when an operator is in-between every pair of operands.

Postfix expression: The expression of the form "a b operator" (ab+) i.e., When every pair of operands is followed by an operator.

Infix to postfix expression conversion procedure:

1. Scan the infix expression from left to right.
2. If the scanned character is an operand, put it in the postfix expression.
3. Otherwise, do the following
 - If the precedence and associativity of the scanned operator are greater than the precedence and associativity of the operator in the stack [or the stack is empty or the stack contains a '('], then push it in the stack. ['^' operator is right associative and other operators like '+', '-', '*', and '/' are left-associative].
 - Check especially for a condition when the operator at the top of the stack and the scanned operator both are '^'. In this condition, the precedence of the scanned operator is higher due to its right associativity. So it will be pushed into the operator stack.
 - In all the other cases when the top of the operator stack is the same as the scanned operator, then pop the operator from the stack because of left associativity due to which the scanned operator has less precedence.
 - Else, Pop all the operators from the stack which are greater than or equal to in precedence than that of the scanned operator.
 - After doing that Push the scanned operator to the stack. (If you encounter parenthesis while popping then stop there and push the scanned operator in the stack.)
4. If the scanned character is a '(', push it to the stack.
5. If the scanned character is a ')', pop the stack and output it until a '(' is encountered, and discard both the parenthesis.

6. Repeat steps 2-5 until the infix expression is scanned.
7. Once the scanning is over, Pop the stack and add the operators in the postfix expression until it is not empty.
8. Finally, print the postfix expression.

Input: A + B * C + D

Output: A B C * + D +

Input: ((A + B) – C * (D / E)) + F

Output: A B + C D E / * - F +

```
import java.util.Stack;

class Conversion
{
    # Write Code Here
}
```

5.5 Reverse a Stack

The stack is a linear data structure which works on the LIFO concept. LIFO stands for last in first out. In the stack, the insertion and deletion are possible at one end the end is called the top of the stack. Define two recursive functions BottomInsertion() and Reverse() to reverse a stack using Python. Define some basic function of the stack like push(), pop(), show(), empty(), for basic operation like respectively append an item in stack, remove an item in stack, display the stack, check the given stack is empty or not.

BottomInsertion(): this method append element at the bottom of the stack and BottomInsertion accept two values as an argument first is stack and the second is elements, this is a recursive method.

Reverse(): the method is reverse elements of the stack, this method accept stack as an argument Reverse() is also a Recursive() function. Reverse() is invoked BottomInsertion() method for completing the reverse operation on the stack.

Input: Elements = [1, 2, 3, 4, 5]

Output: Original Stack

5
4
3
2
1

Stack after Reversing

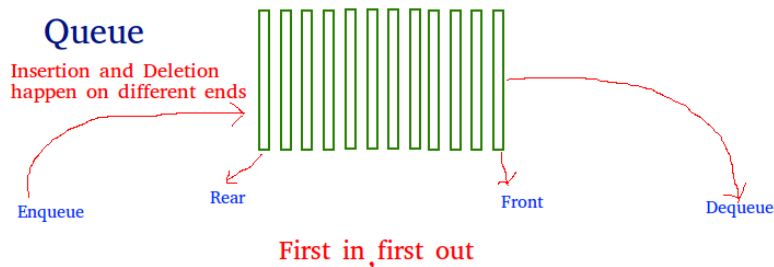
1
2
3
4
5

```
import java.util.Stack;
class StackClass {
    # Write Code Here
}
```

6. Queue

6.1 Linear Queue

Linear queue is a linear data structure that stores items in First in First out (FIFO) manner. With a queue the least recently added item is removed first. A good example of queue is any queue of consumers for a resource where the consumer that came first is served first.



```
import java.util.Scanner;

public class LinearQueue
{
    # Write Code Here
}

public static boolean isEmpty() {
    return front == rear;
}

public static boolean isFull() {
    return rear == MAX;
}

public static void enqueue(int item)
{
    # Write Code Here
}

public static void dequeue()
{
    # Write Code Here
}

public static void display()
{
    # Write Code Here
}

public static void main(String[] args)
{
    # Write Code Here
}
}
```

6.2 Stack using Queues

Implement a last-in-first-out (LIFO) stack using only two queues. The implemented stack should support all the functions of a normal stack (push, top, pop, and empty).

- void push(int x) Pushes element x to the top of the stack.
- int pop() Removes the element on the top of the stack and returns it.
- int top() Returns the element on the top of the stack.
- boolean empty() Returns true if the stack is empty, false otherwise.

Input:

["MyStack", "push", "push", "top", "pop", "empty"]

[[], [1], [2], [], [], []]

Output:

```
[null, null, null, 2, 2, false]
```

```
import java.util.LinkedList;
import java.util.Queue;

class MyStack
{
    # Write Code Here
}

public void push(int x)
{
    # Write Code Here
}

public int pop()
{
    return queue.remove();
}

public int top() {
    return queue.peek();
}

public boolean empty() {
    return queue.isEmpty();
}

public static void main(String[] args)
{
    # Write Code Here
}
```

6.3 Queue using Stacks

Implement a first in first out (FIFO) queue using only two stacks. The implemented queue should support all the functions of a normal queue (push, peek, pop, and empty).

- void push(int x) Pushes element x to the back of the queue.
- int pop() Removes the element from the front of the queue and returns it.
- int peek() Returns the element at the front of the queue.
- boolean empty() Returns true if the queue is empty, false otherwise.

Input:

```
["MyQueue", "push", "push", "peek", "pop", "empty"]
[[], [1], [2], [], [], []]
```

Output:

```
[null, null, null, 1, 1, false]
```

```
import java.util.Stack;

class MyQueue {

    private Stack<Integer> stack1;
    private Stack<Integer> stack2;
    public MyQueue() {
        stack1 = new Stack<>();
        stack2 = new Stack<>();
    }
}
```

```

public void push(int x) {
    stack1.push(x);
}
public int pop()
{
    # Write Code Here
}
public int peek()
{
    # Write Code Here
}
public boolean empty() {
    return stack1.isEmpty() && stack2.isEmpty();
}
public static void main(String[] args)
{
    # Write Code Here
}
}

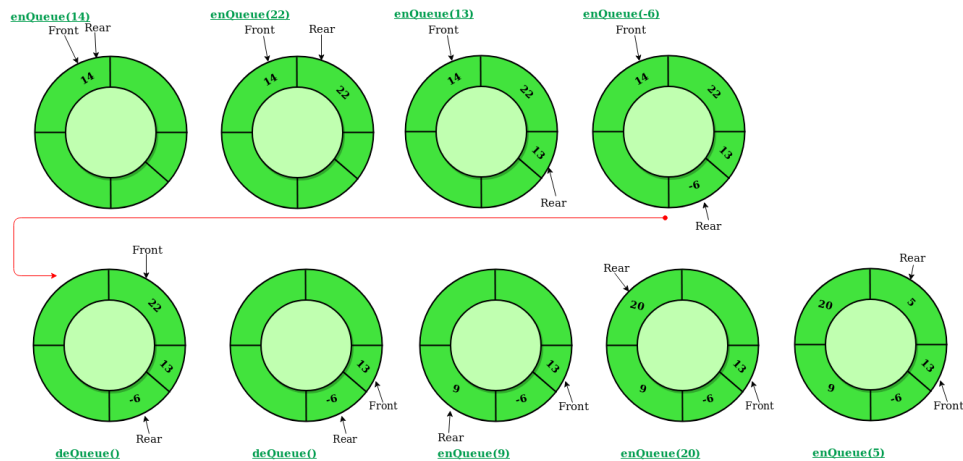
```

6.4 Circular Queue

A Circular Queue is an extended version of a normal queue where the last element of the queue is connected to the first element of the queue forming a circle. The operations are performed based on FIFO (First In First Out) principle. It is also called 'Ring Buffer'.

Operations on Circular Queue:

- **Front:** Get the front item from the queue.
- **Rear:** Get the last item from the queue.
- **enQueue(value)** This function is used to insert an element into the circular queue. In a circular queue, the new element is always inserted at the rear position.
 - Check whether the queue is full – [i.e., the rear end is in just before the front end in a circular manner].
 - If it is full then display Queue is full.
 - If the queue is not full then, insert an element at the end of the queue.
- **deQueue()** This function is used to delete an element from the circular queue. In a circular queue, the element is always deleted from the front position.
 - Check whether the queue is Empty.
 - If it is empty then display Queue is empty.
 - If the queue is not empty, then get the last element and remove it from the queue.



Implement Circular Queue using Array:

5. Initialize an array queue of size **n**, where **n** is the maximum number of elements that the queue can hold.
6. Initialize two variables **front** and **rear** to -1.
7. **Enqueue:** To enqueue an element **x** into the queue, do the following:
 - Increment **rear** by 1.
 - If **rear** is equal to **n**, set **rear** to 0.
 - If **front** is -1, set **front** to 0.
 - Set **queue[rear]** to **x**.
8. **Dequeue:** To dequeue an element from the queue, do the following:
 - Check if the queue is empty by checking if **front** is -1.
 - If it is, return an error message indicating that the queue is empty.
 - Set **x** to **queue[front]**.
 - If **front** is equal to **rear**, set **front** and **rear** to -1.
 - Otherwise, increment **front** by 1 and if **front** is equal to **n**, set **front** to 0.
 - Return **x**.

```
class CircularQueue {

    private int size;
    private int front, rear;
    private int[] queue;

    public CircularQueue(int size) {
        this.size = size;
        this.queue = new int[size];
        this.front = this.rear = -1;
    }

    public void enqueue(int data)
    {
        # Write Code Here
    }

    public int dequeue()
    {

```

```

        # Write Code Here
    }
    public void display()
    {
        # Write Code Here
    }
    public static void main(String[] args)
    {
        # Write Code Here
    }
}

```

6.5 Deque (Doubly Ended Queue)

In a Deque (Doubly Ended Queue), one can perform insert (append) and delete (pop) operations from both the ends of the container. There are two types of Deque:

1. **Input Restricted Deque:** Input is limited at one end while deletion is permitted at both ends.
2. **Output Restricted Deque:** Output is limited at one end but insertion is permitted at both ends.

Operations on Deque:

1. **append():** This function is used to insert the value in its argument to the right end of the deque.
2. **appendleft():** This function is used to insert the value in its argument to the left end of the deque.
3. **pop():** This function is used to delete an argument from the right end of the deque.
4. **popleft():** This function is used to delete an argument from the left end of the deque.
5. **index(ele, beg, end):** This function returns the first index of the value mentioned in arguments, starting searching from beg till end index.
6. **insert(i, a):** This function inserts the value mentioned in arguments(a) at index(i) specified in arguments.
7. **remove():** This function removes the first occurrence of the value mentioned in arguments.
8. **count():** This function counts the number of occurrences of value mentioned in arguments.
9. **len(dequeue):** Return the current size of the dequeue.
10. **Deque[0]:** We can access the front element of the deque using indexing with de[0].
11. **Deque[-1]:** We can access the back element of the deque using indexing with de[-1].
12. **extend(iterable):** This function is used to add multiple values at the right end of the deque. The argument passed is iterable.
13. **extendleft(iterable):** This function is used to add multiple values at the left end of the deque. The argument passed is iterable. Order is reversed as a result of left appends.
14. **reverse():** This function is used to reverse the order of deque elements.
15. **rotate():** This function rotates the deque by the number specified in arguments. If the number specified is negative, rotation occurs to the left. Else rotation is to right.

```

import java.util.ArrayDeque;
import java.util.Deque;

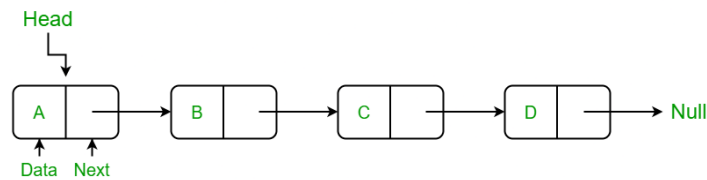
public class DequeOperations
{
    # Write Code Here
}

```

7. Linked List

7.1 Singly Linked List

A singly linked list is a linear data structure in which the elements are not stored in contiguous memory locations and each element is connected only to its next element using a pointer.



Creating a linked list involves the following operations:

1. Creating a Node class:
2. Insertion at beginning:
3. Insertion at end
4. Insertion at middle
5. Update the node
6. Deletion at beginning
7. Deletion at end
8. Deletion at middle
9. Remove last node
10. Linked list traversal
11. Get length

```
class Node {
    String data;
    Node next;

    Node(String data) {
        this.data = data;
        this.next = null;
    }
}

class LinkedList
{
    # Write Code Here
}

public void insertAtEnd(String data)
{
    # Write Code Here
}

public void updateNode(String val, int index)
{
    # Write Code Here
}

public void remove_first_node() {
    # Write Code Here
}

public void remove_last_node()
{
    # Write Code Here
}

public void remove_at_index(int index)
{
    # Write Code Here
}
```

```

    }

    public void remove_node(String data)
    {
        # Write Code Here
    }

    public int sizeOfLL()
    {
        # Write Code Here
    }

    public void printLL()
    {
        # Write Code Here
    }

    public static void main(String[] args)
    {
        # Write Code Here
    }
}

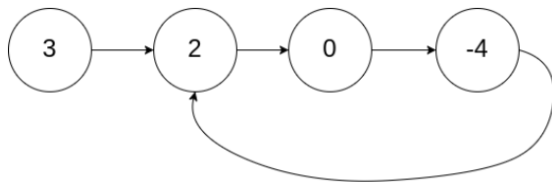
```

7.2 Linked List Cycle

Given head, the head of a linked list, determine if the linked list has a cycle in it. There is a cycle in a linked list if there is some node in the list that can be reached again by continuously following the next pointer. Internally, pos is used to denote the index of the node that tail's next pointer is connected to.

Note that pos is not passed as a parameter.

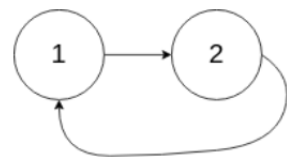
Return true if there is a cycle in the linked list. Otherwise, return false.



Input: head = [3, 2, 0, -4], pos = 1

Output: true

Explanation: There is a cycle in the linked list, where the tail connects to the 1st node (0-indexed).



Input: head = [1, 2], pos = 0

Output: true

Explanation: There is a cycle in the linked list, where the tail connects to the 0th node.



Input: head = [1], pos = -1

Output: false

Explanation: There is no cycle in the linked list.

```
class ListNode
```

```

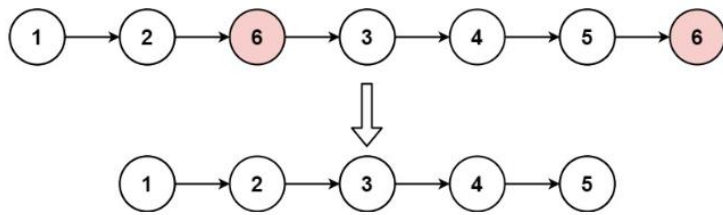
    {
        # Write Code Here
    }

public class Solution
{
    # Write Code Here
}

```

7.3 Remove Linked List Elements

Given the head of a linked list and an integer val, remove all the nodes of the linked list that has Node.val == val, and return the new head.



Input: head = [1, 2, 6, 3, 4, 5, 6], val = 6

Output: [1, 2, 3, 4, 5]

Input: head = [], val = 1

Output: []

Input: head = [7, 7, 7, 7], val = 7

Output: []

```
class ListNode {
    # Write Code Here
}

public class Solution {
    public boolean hasCycle(ListNode head)
    {
        # Write Code Here
    }

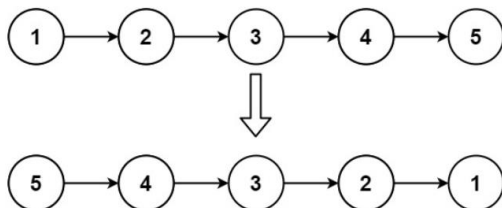
    public static void main(String[] args)
    {
        # Write Code Here
    }
}
```

7.4 Reverse Linked List

Given the head of a singly linked list, reverse the list, and return the reversed list.

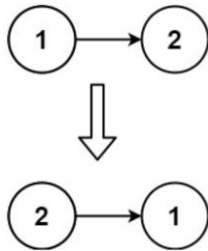
Input: head = [1, 2, 3, 4, 5]

Output: [5, 4, 3, 2, 1]



Input: head = [1, 2]

Output: [2, 1]



```
class ListNode {
    int val;
    ListNode next;

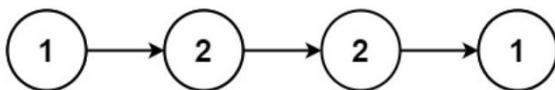
    ListNode(int val) {
        this.val = val;
        this.next = null;
    }
}

public class Solution {
    public ListNode reverseList(ListNode head)
    {
        # Write Code Here
    }

    public static void main(String[] args)
    {
        # Write Code Here
    }
}
```

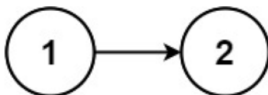
7.5 Palindrome Linked List

Given the head of a singly linked list, return true if it is a palindrome or false otherwise.



Input: head = [1, 2, 2, 1]

Output: true



Input: head = [1, 2]

Output: false

```
class ListNode
{
    # Write Code Here
}

public class Solution {
    public boolean isPalindrome(ListNode head)
    {
        # Write Code Here
    }
}
```

```

    }

    public static void main(String[] args)
    {
        # Write Code Here
    }
}

```

7.6 Middle of the Linked List

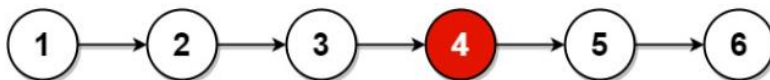
Given the head of a singly linked list, return the middle node of the linked list. If there are two middle nodes, return the second middle node.



Input: head = [1, 2, 3, 4, 5]

Output: [3, 4, 5]

Explanation: The middle node of the list is node 3.



Input: head = [1, 2, 3, 4, 5, 6]

Output: [4, 5, 6]

Explanation: Since the list has two middle nodes with values 3 and 4, we return the second one.

```

class ListNode
{
    # Write Code Here
}

public class Solution
{
    # Write Code Here

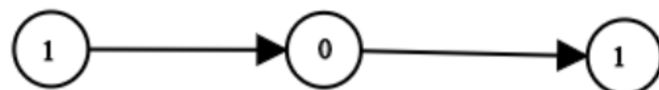
    public static void main(String[] args) {
        # Write Code Here
    }
}

```

7.7 Convert Binary Number in a Linked List to Integer

Given head which is a reference node to a singly-linked list. The value of each node in the linked list is either 0 or 1. The linked list holds the binary representation of a number.

Return the decimal value of the number in the linked list. The most significant bit is at the head of the linked list.



Input: head = [1, 0, 1]

Output: 5

Explanation: (101) in base 2 = (5) in base 10

Input: head = [0]

Output: 0

```

class ListNode {
    int val;
    ListNode next;

    ListNode(int val) {
        this.val = val;
        this.next = null;
    }
}

public class Solution
{
    # Write Code Here
}

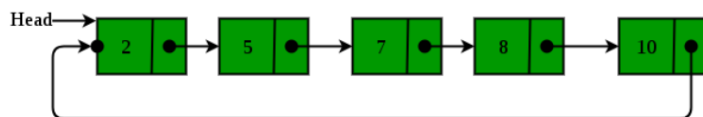
public static void main(String[] args)
{
    # Write Code Here
}
}

```

8. Circular Single Linked List and Doubly Linked List

8.1 Circular Linked List

The circular linked list is a linked list where all nodes are connected to form a circle. In a circular linked list, the first node and the last node are connected to each other which forms a circle. There is no NULL at the end.



Operations on the circular linked list:

1. Insertion at the beginning
2. Insertion at the end
3. Insertion in between the nodes
4. Deletion at the beginning
5. Deletion at the end
6. Deletion in between the nodes
7. Traversal

```

import java.util.ArrayList;
public class Main{
    static class Node{
        int data;
        Node next;
        Node(int data){
            this.data = data;
            this.next = null;
        }
    }
    static class CircularLinkedList
    {

```

```

    # Write Code Here
}
Node addAfter(int data, int item)
{
    # Write Code Here
}
void deleteNode(Node last, int key)
{
    # Write Code Here
}

```

System.

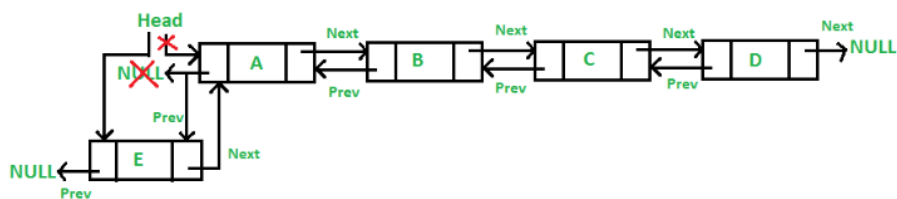
8.2 Doubly Linked List

The A doubly linked list is a type of linked list in which each node consists of 3 components:

1. *prev - address of the previous node
2. data - data item
3. *next - address of next node.



Double Linked List Node



Operations on the Double Linked List:

1. Insertion at the beginning
2. Insertion at the end
3. Insertion in between the nodes
4. Deletion at the beginning
5. Deletion at the end
6. Deletion in between the nodes
7. Traversal

```
import java.util.Scanner;
```

```

class Node {
    int data;
    Node next;
    Node prev;

    Node(int data) {
        this.data = data;
        this.next = null;
        this.prev = null;
    }
}

```

```

class DLinkedList {
    Node head;
}

```

```

int ctr;

DLinkedList() {
    this.head = null;
    this.ctr = 0;
}

void insertBeg(int data)
{
    # Write Code Here
}
void insertEnd(int data)
{
    # Write Code Here
}

void deleteBeg()
{
    # Write Code Here
}
void deleteEnd()
{
    # Write Code Here
}

void insertPos(int pos, int data)
{
    # Write Code Here
}

void deletePos(int pos)
{
    # Write Code Here
}

void traverseF()
{
    # Write Code Here
}

void traverseR()
{
    # Write Code Here
}

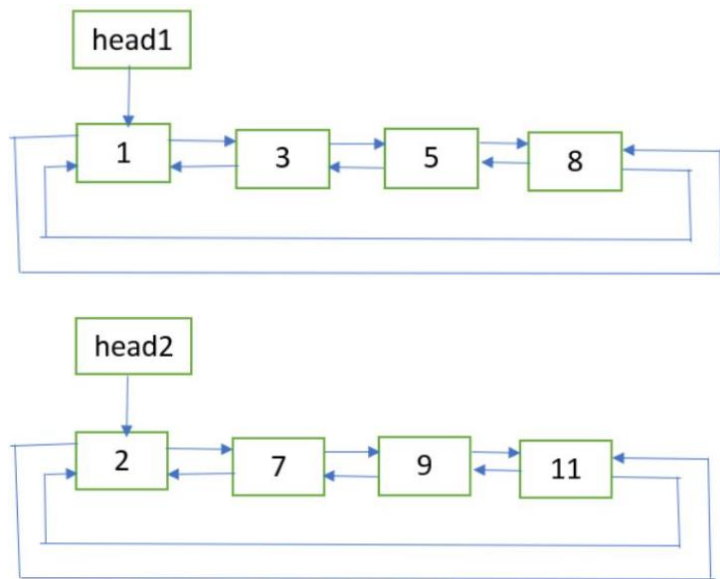
public class Main {
    public static void main(String[] args)
    {
        # Write Code Here
    }
}

```

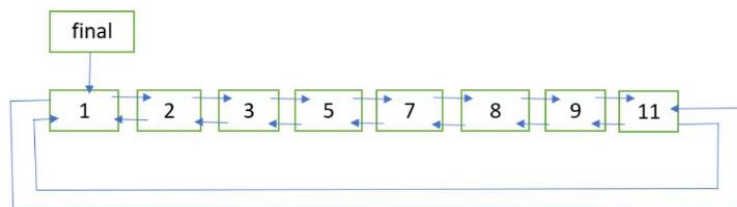
8.3 Sorted Merge of Two Sorted Doubly Circular Linked Lists

Given two sorted Doubly circular Linked List containing n_1 and n_2 nodes respectively. The problem is to merge the two lists such that resultant list is also in sorted order.

Input: List 1 and List 2



Output: Merged List



Procedure for Merging Doubly Linked List:

1. If `head1 == NULL`, return `head2`.
2. If `head2 == NULL`, return `head1`.
3. Let **last1** and **last2** be the last nodes of the two lists respectively. They can be obtained with the help of the previous links of the first nodes.
4. Get pointer to the node which will be the last node of the final list. If `last1.data < last2.data`, then **last_node** = `last2`, Else **last_node** = `last1`.
5. Update `last1.next = last2.next = NULL`.
6. Now merge the two lists as two sorted doubly linked list are being merged.
Refer **merge** procedure of this post. Let the first node of the final list be **finalHead**.
7. Update `finalHead.prev = last_node` and `last_node.next = finalHead`.
8. Return **finalHead**.

```
class Node {
    int data;
    Node next, prev;

    Node(int data) {
        this.data = data;
        this.next = null;
        this.prev = null;
    }
}

public class SortedMergeDoublyCircularLinkedList
{
    # Write Code Here
}
```

```

static Node mergeUtil(Node head1, Node head2)
{
    # Write Code Here
}

static void printList(Node head)
{
    # Write Code Here
}

public static void main(String[] args)
{
    # Write Code Here
}
}

```

8.4 Delete all occurrences of a given key in a Doubly Linked List

Given a doubly linked list and a key x. The problem is to delete all occurrences of the given key x from the doubly linked list.

Input: 2 <-> 2 <-> 10 <-> 8 <-> 4 <-> 2 <-> 5 <-> 2

x = 2

Output: 10 <-> 8 <-> 4 <-> 5

Algorithm:

delAllOccurOfGivenKey (head_ref, x)

if head_ref == NULL

return

Initialize **current** = head_ref

Declare **next**

while current != NULL

if current->data == x

next = current->next

deleteNode(head_ref, current)

current = next

else

current = current->next

```

class Node {
    int data;
    Node next, prev;

    Node(int data) {
        this.data = data;
        this.next = null;
        this.prev = null;
    }
}

public class DeleteOccurrenceInDoublyLinkedList
{
    # Write Code Here
}

static Node deleteAllOccurOfX(Node head, int x)
{
    # Write Code Here
}

```

```

    }

    static void printList(Node head)
    {
        # Write Code Here
    }

    public static void main(String[] args)
    {
        # Write Code Here
    }
}

```

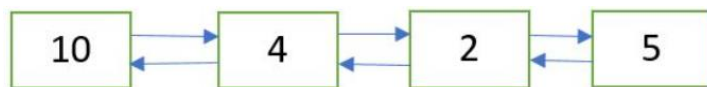
8.5 Delete a Doubly Linked List Node at a Given Position

Given a doubly linked list and a position n. The task is to delete the node at the given position n from the beginning.

Input: Initial doubly linked list



Output: Doubly Linked List after deletion of node at position n = 2



Procedure:

1. Get the pointer to the node at position n by traversing the doubly linked list up to the nth node from the beginning.
2. Delete the node using the pointer obtained in Step 1.

```

class Node {
    int data;
    Node next, prev;

    Node(int data) {
        this.data = data;
        this.next = null;
        this.prev = null;
    }
}

public class DeleteNodeAtGivenPosition
{
    # Write Code Here
}

static Node deleteNode(Node head, Node del)
{
    # Write Code Here
}

static Node deleteNodeAtGivenPos(Node head, int n)
{
    # Write Code Here
}

```

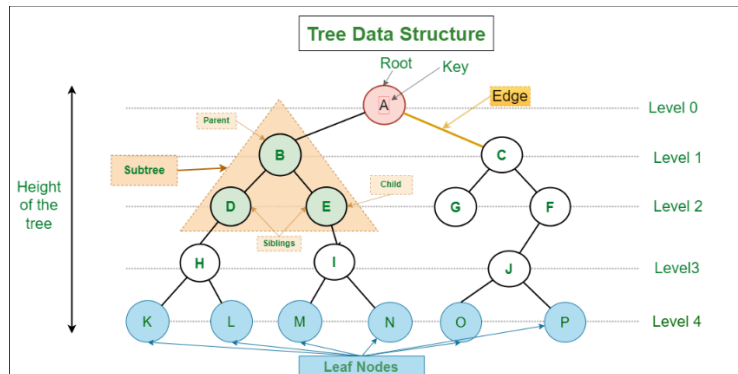


```
static void printList(Node head)
{
    # Write Code Here
}
}
```

9. Trees

9.1 Tree Creation and Basic Tree Terminologies

A tree data structure is a hierarchical structure that is used to represent and organize data in a way that is easy to navigate and search. It is a collection of nodes that are connected by edges and has a hierarchical relationship between the nodes.



Basic Terminologies in Tree:

1. **Parent Node:** The node which is a predecessor of a node is called the parent node of that node. {B} is the parent node of {D, E}.
2. **Child Node:** The node which is the immediate successor of a node is called the child node of that node. Examples: {D, E} are the child nodes of {B}.
3. **Root Node:** The topmost node of a tree or the node which does not have any parent node is called the root node. {A} is the root node of the tree. A non-empty tree must contain exactly one root node and exactly one path from the root to all other nodes of the tree.
4. **Leaf Node or External Node:** The nodes which do not have any child nodes are called leaf nodes. {K, L, M, N, O, P} are the leaf nodes of the tree.
5. **Ancestor of a Node:** Any predecessor nodes on the path of the root to that node are called Ancestors of that node. {A, B} are the ancestor nodes of the node {E}
6. **Descendant:** Any successor node on the path from the leaf node to that node. {E, I} are the descendants of the node {B}.
7. **Sibling:** Children of the same parent node are called siblings. {D, E} are called siblings.
8. **Level of a node:** The count of edges on the path from the root node to that node. The root node has level 0.
9. **Internal node:** A node with at least one child is called Internal Node.
10. **Neighbour of a Node:** Parent or child nodes of that node are called neighbors of that node.
11. **Subtree:** Any node of the tree along with its descendant.

```
import java.util.ArrayList;
import java.util.List;

public class TreeBasicTerminologies
{
    # Write Code Here
}

static void printChildren(int root, List<List<Integer>> adj)
{
    # Write Code Here
}

static void printLeafNodes(int root, List<List<Integer>> adj)
```

```

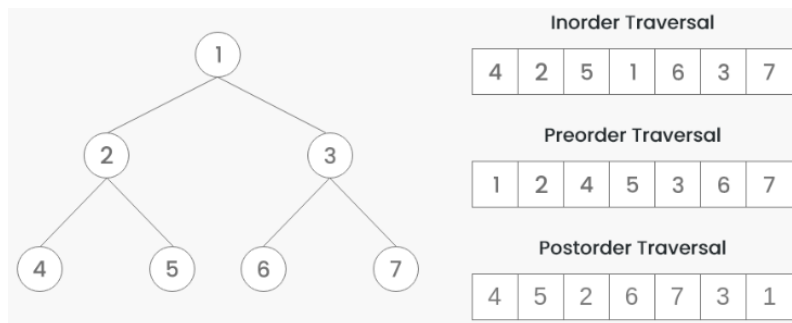
    {
        # Write Code Here
    }
    static void printDegrees(int root, List<List<Integer>> adj)
    {
        # Write Code Here
    }
    public static void main(String[] args)
    {
        # Write Code Here
    }
}

```

9.2 Binary Tree Traversal Techniques

A binary tree data structure can be traversed in following ways:

1. Inorder Traversal
2. Preorder Traversal
3. Postorder Traversal
4. Level Order Traversal



Algorithm Inorder (tree)

1. Traverse the left subtree, i.e., call Inorder(left->subtree)
2. Visit the root.
3. Traverse the right subtree, i.e., call Inorder(right->subtree)

Algorithm Preorder (tree)

1. Visit the root.
2. Traverse the left subtree, i.e., call Preorder(left->subtree)
3. Traverse the right subtree, i.e., call Preorder(right->subtree)

Algorithm Postorder (tree)

1. Traverse the left subtree, i.e., call Postorder(left->subtree)
2. Traverse the right subtree, i.e., call Postorder(right->subtree)
3. Visit the root.

```

import java.util.Scanner;
class Node
{
    # Write Code Here
}
class BT {
    Node root;
    BT() {

```

```

        this.root = null;
    }
    void insert(int data)
    {
        # Write Code Here
    }
    Node insertRec(Node root, int data)
    {
        # Write Code Here
    }
    void postorder(Node root)
    {
        # Write Code Here
    }
    void preorder(Node root)
    {
        # Write Code Here
    }
    }
    void inorder(Node root)
    {
        # Write Code Here
    }
}

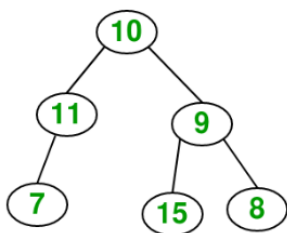
public class BinaryTreeTraversal {
    public static void main(String[] args)
    {
        # Write Code Here
    }
}

```

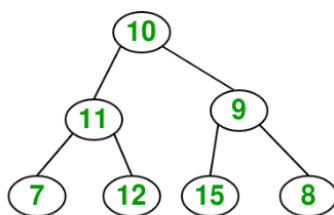
9.3 Insertion in a Binary Tree in Level Order

Given a binary tree and a key, insert the key into the binary tree at the first position available in level order.

Input: Consider the tree given below



Output:



After inserting 12

The idea is to do an iterative level order traversal of the given tree using queue. If we find a node whose left child is empty, we make a new key as the left child of the node. Else if we find a node whose right child is empty, we make the new key as the right child. We keep traversing the tree until we find a node whose either left or right child is empty.

```
class Node
{
# Write Code Here
}

public class BinaryTreeInsertion
{
# Write Code Here
}

static Node insert(Node root, int key)
{
# Write Code Here
}

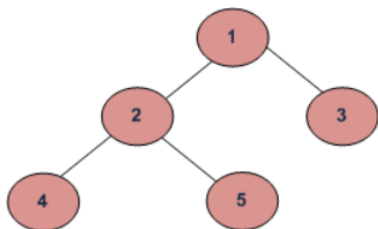
public static void main(String[] args)
{
# Write Code Here
}
```

9.4 Finding the Maximum Height or Depth of a Binary Tree

Given a binary tree, the task is to find the height of the tree. The height of the tree is the number of edges in the tree from the root to the deepest node.

Note: The height of an empty tree is 0.

Input: Consider the tree below



Recursively calculate the height of the left and the right subtrees of a node and assign height to the node as max of the heights of two children plus 1.

$\text{maxDepth}('1') = \max(\text{maxDepth}('2'), \text{maxDepth}('3')) + 1 = 2 + 1$

because recursively

$\text{maxDepth}('2') = \max(\text{maxDepth}('4'), \text{maxDepth}('5')) + 1 = 1 + 1$ and (as height of both '4' and '5' are 1)

$\text{maxDepth}('3') = 1$

Procedure:

- Recursively do a Depth-first search.
- If the tree is empty then return 0
- Otherwise, do the following
 - Get the max depth of the left subtree recursively i.e. call $\text{maxDepth}(\text{tree} \rightarrow \text{left-subtree})$
 - Get the max depth of the right subtree recursively i.e. call $\text{maxDepth}(\text{tree} \rightarrow \text{right-subtree})$
 - Get the max of max depths of left and right subtrees and add 1 to it for the current node.

$$\text{maxDepth} = \max(\text{maxDepth of left subtree}, \text{maxDepth of right subtree}) + 1$$

- Return max_depth.

```
class Node
{
    int data;
    Node left, right;

    Node(int data) {
        this.data = data;
        this.left = null;
        this.right = null;
    }
}

public class MaximumDepthOfTree
{
    # Write Code Here
}

public static void main(String[] args)
{
    # Write Code Here
}
```

9.5 Deletion in a Binary Tree

Given a binary tree, delete a node from it by making sure that the tree shrinks from the bottom (i.e. the deleted node is replaced by the bottom-most and rightmost node).

Input: Delete 10 in below tree

```

    10
   /  \
  20   30
```

Output:

```

    30
   /
  20
```

Input: Delete 20 in below tree

```

    10
   /  \
  20   30
   \
   40
```

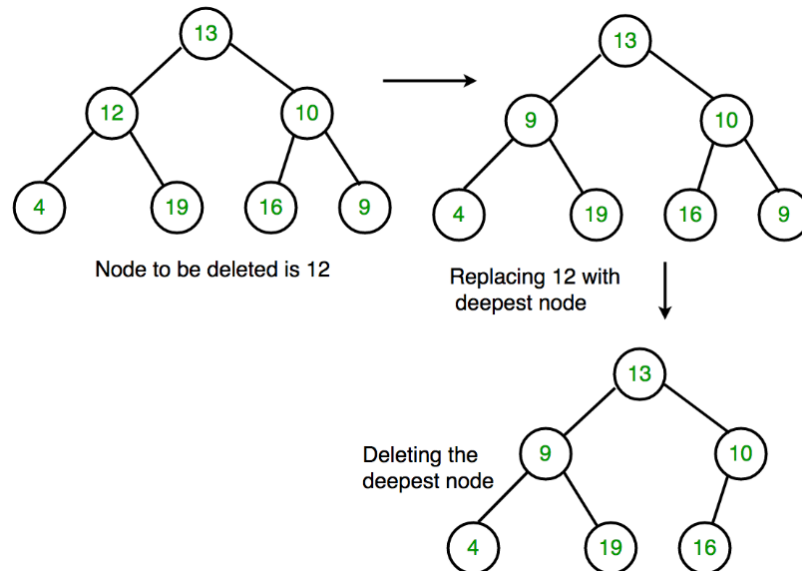
Output:

```

    10
   /  \
  40   30
```

Algorithm:

1. Starting at the root, find the deepest and rightmost node in the binary tree and the node which we want to delete.
2. Replace the deepest rightmost node's data with the node to be deleted.
3. Then delete the deepest rightmost node.



```
class Node {
    int data;
    Node left, right;

    Node(int data) {
        this.data = data;
        this.left = null;
        this.right = null;
    }
}

public class BinaryTreeDeletion
{
    # Write Code Here
}

static void deleteDeepest(Node root, Node dNode)
{
    # Write Code Here
}

static Node deletion(Node root, int key)
{
    # Write Code Here
}

public static void main(String[] args)
{
    # Write Code Here
}
}
```

10. Binary Search Tree (BST)

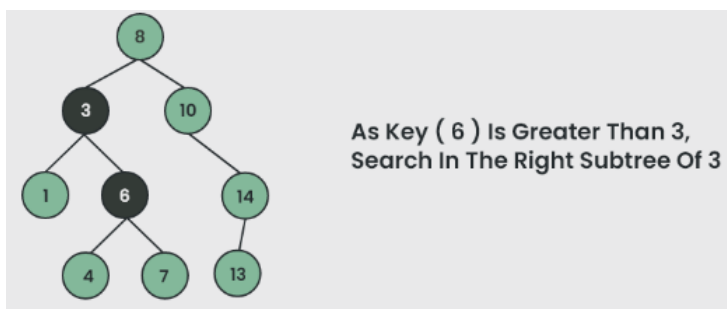
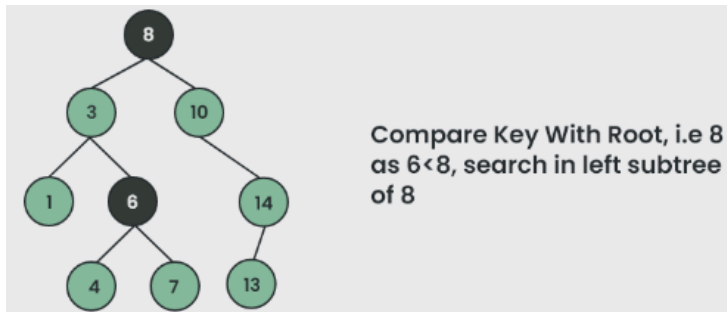
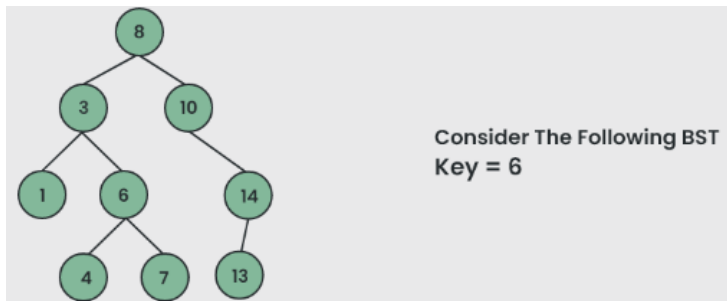
10.1 Searching in Binary Search Tree

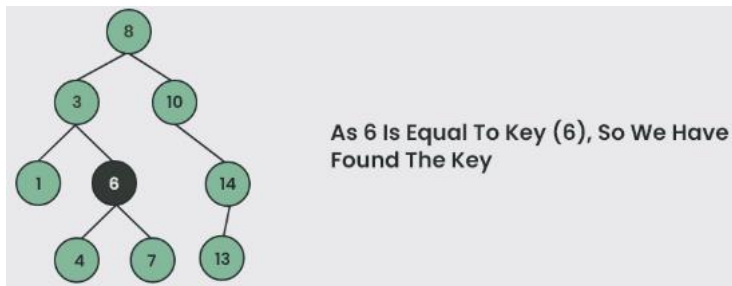
Given a BST, the task is to delete a node in this BST. For searching a value in BST, consider it as a sorted array. Perform search operation in BST using Binary Search Algorithm.

Algorithm to search for a key in a given Binary Search Tree:

Let's say we want to search for the number **X**, We start at the root. Then:

- We compare the value to be searched with the value of the root.
 - If it's equal we are done with the search if it's smaller we know that we need to go to the left subtree because in a binary search tree all the elements in the left subtree are smaller and all the elements in the right subtree are larger.
- Repeat the above step till no more traversal is possible
- If at any iteration, key is found, return True. Else False.





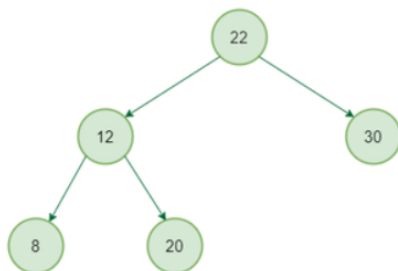
```
// Node class to represent each node of the BST
class Node {
    int key;
    Node left, right;

    public Node(int item) {
        key = item;
        left = right = null;
    }
}
class BST
{
    # Write Code Here
}
    Node search(int key) {
        return searchRec(root, key);
    }
    Node searchRec(Node root, int key)
    {
        # Write Code Here
    }
    public static void main(String[] args)
    {
        # Write Code Here
    }
}
```

10.2 Find the node with Minimum Value in a BST

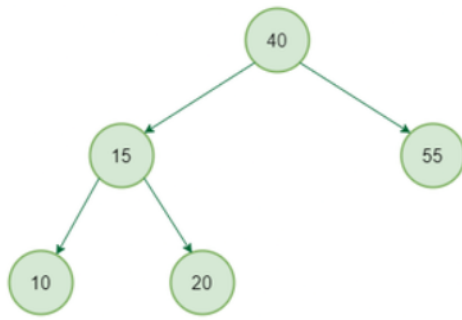
Write a function to find the node with minimum value in a Binary Search Tree.

Input: Consider the tree given below



Output: 8

Input: Consider the tree given below



Output: 10

```
import java.util.ArrayList;
import java.util.List;
```

```
class Node {
    int data;
    Node left, right;

    public Node(int item) {
        data = item;
        left = right = null;
    }
}
```

```
class BinarySearchTree
{
    # Write Code Here
}

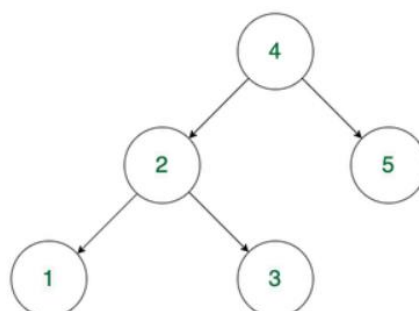
public static void main(String[] args)
{
    # Write Code Here
}
```

10.3 Check if a Binary Tree is BST or not

A binary search tree (BST) is a node-based binary tree data structure that has the following properties.

1. The left subtree of a node contains only nodes with keys less than the node's key.
2. The right subtree of a node contains only nodes with keys greater than the node's key.
3. Both the left and right subtrees must also be binary search trees.
4. Each node (item in the tree) has a distinct key.

Input: Consider the tree given below



Output: Check if max value in left subtree is smaller than the node and min value in right subtree greater than the node, then print it "Is BST" otherwise "Not a BST"

Procedure:

1. If the current node is null then return true
2. If the value of the left child of the node is greater than or equal to the current node then return false
3. If the value of the right child of the node is less than or equal to the current node then return false
4. If the left subtree or the right subtree is not a BST then return false
5. Else return true

```
class Node {
    int data;
    Node left, right;

    public Node(int item) {
        data = item;
        left = right = null;
    }
}

class BinaryTree
{
    # Write Code Here
}

boolean isBST(Node node) {
    return isBSTUtil(node, Integer.MIN_VALUE, Integer.MAX_VALUE);
}

boolean isBSTUtil(Node node, int min, int max)
{
    # Write Code Here
}

public static void main(String[] args)
{
    # Write Code Here
}
}
```

10.4 Second Largest Element in BST

Given a Binary search tree (BST), find the second largest element.

Input: Root of below BST

```
    10
   /
  5
```

Output: 5

Input: Root of below BST

```
    10
   / \
```

```

5   20
   \
    30

```

Output: 20

Procedure: The second largest element is second last element in inorder traversal and second element in reverse inorder traversal. We traverse given Binary Search Tree in reverse inorder and keep track of counts of nodes visited. Once the count becomes 2, we print the node.

```

class Node {
    int key;
    Node left, right;

    public Node(int item) {
        key = item;
        left = right = null;
    }
}

class BinarySearchTree
{
    # Write Code Here

    secondLargestUtil(node.right);
    count++;

    // If count is equal to 2 then this is the second largest
    if (count == 2) {
        System.out.println("The second largest element is " + node.key);
        return;
    }

    secondLargestUtil(node.left);
}

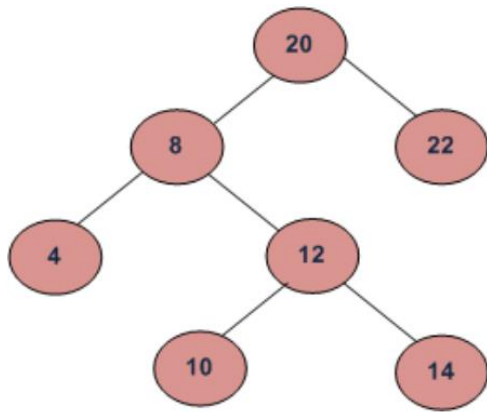
// Function to find the second largest element
void secondLargest(Node node) {
    count = 0;
    secondLargestUtil(node);
}

// Driver code
public static void main(String[] args)
{
    # Write Code Here
}

```

Try:

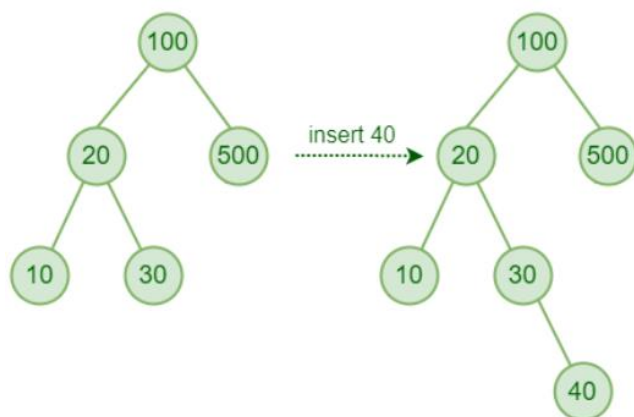
1. **Kth largest element in BST when modification to BST is not allowed:** Given a Binary Search Tree (BST) and a positive integer k, find the k'th largest element in the Binary Search Tree. For a given BST, if k = 3, then output should be 14, and if k = 5, then output should be 10.



10.5 Insertion in Binary Search Tree (BST)

Given a Binary search tree (BST), the task is to insert a new node in this BST.

Input: Consider a BST and insert the element 40 into it.



Procedure for inserting a value in a BST:

A new key is always inserted at the leaf by maintaining the property of the binary search tree. We start searching for a key from the root until we hit a leaf node. Once a leaf node is found, the new node is added as a child of the leaf node. The below steps are followed while we try to insert a node into a binary search tree:

- Check the value to be inserted (say X) with the value of the current node (say val) we are in:
 - If X is less than val move to the left subtree.
 - Otherwise, move to the right subtree.
- Once the leaf node is reached, insert X to its right or left based on the relation between X and the leaf node's value.

```
// A utility class that represents an individual node in a BST
class Node {
    int val;
    Node left, right;

    public Node(int item) {
        val = item;
        left = right = null;
    }
}
```

```

}

class BinarySearchTree
{
    # Write Code Here

}

void inorder()
{
    inorderRec(root);
}

void inorderRec(Node root)
{
    # Write Code Here

}

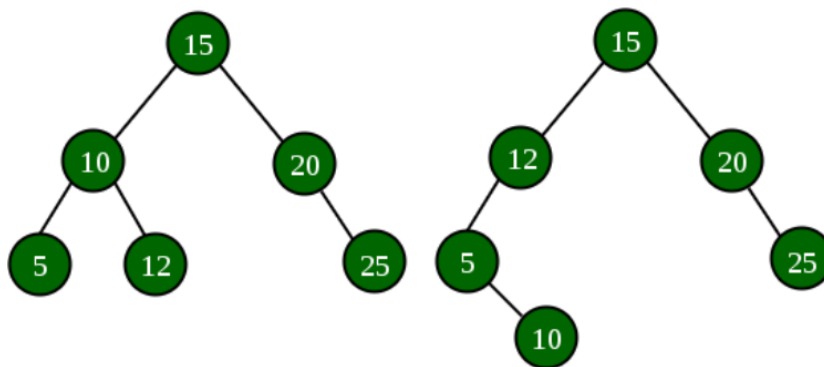
// Driver code
public static void main(String[] args)
{
    # Write Code Here
}
}

```

Try:

1. **Check if two BSTs contain same set of elements:** Given two Binary Search Trees consisting of unique positive elements, we have to check whether the two BSTs contain the same set of elements or not.

Input: Consider two BSTs which contains same set of elements {5, 10, 12, 15, 20, 25}, but the structure of the two given BSTs can be different.



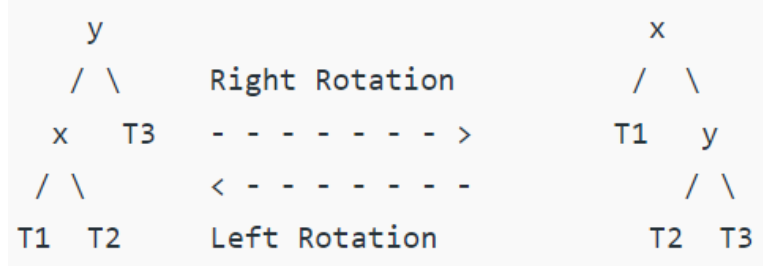
11. AVL Tree

11.1 Insertion in an AVL Tree

AVL tree is a self-balancing Binary Search Tree (BST) where the difference between heights of left and right subtrees cannot be more than one for all nodes. To make sure that the given tree remains AVL after every insertion, we must augment the standard BST insert operation to perform some re-balancing. Following are two basic operations that can be performed to balance a BST without violating the BST property ($\text{keys}(\text{left}) < \text{key}(\text{root}) < \text{keys}(\text{right})$).

- Left Rotation
- Right Rotation

T1, T2 and T3 are subtrees of the tree, rooted with y (on the left side) or x (on the right side)



Keys in both of the above trees follow the following order

$\text{keys}(T1) < \text{key}(x) < \text{keys}(T2) < \text{key}(y) < \text{keys}(T3)$

So BST property is not violated anywhere.

Procedure for inserting a node into an AVL tree

Let the newly inserted node be w

- Perform standard BST insert for w.
- Starting from w, travel up and find the first unbalanced node. Let z be the first unbalanced node, y be the child of z that comes on the path from w to z and x be the grandchild of z that comes on the path from w to z.
- Re-balance the tree by performing appropriate rotations on the subtree rooted with z. There can be 4 possible cases that need to be handled as x, y and z can be arranged in 4 ways.
- Following are the possible 4 arrangements:
 - y is the left child of z and x is the left child of y (Left Left Case)
 - y is the left child of z and x is the right child of y (Left Right Case)
 - y is the right child of z and x is the right child of y (Right Right Case)
 - y is the right child of z and x is the left child of y (Right Left Case)

```
class TreeNode {
    int val, height;
    TreeNode left, right;

    TreeNode(int d) {
        val = d;
        height = 1;
    }
}

class AVL_Tree {

    # write the code

}

TreeNode leftRotate(TreeNode x)
{
    # write the code
}

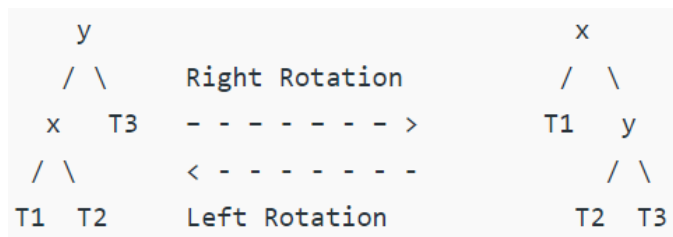
public static void main(String[] args) {
    # write the code
}
```

11.2 Deletion in an AVL Tree

Given an AVL tree, make sure that the given tree remains AVL after every deletion, we must augment the standard BST delete operation to perform some re-balancing. Following are two basic operations that can be performed to re-balance a BST without violating the BST property ($\text{keys}(\text{left}) < \text{key}(\text{root}) < \text{keys}(\text{right})$).

1. Left Rotation
2. Right Rotation

T1, T2 and T3 are subtrees of the tree rooted with y (on left side)
or x (on right side)



Keys in both of the above trees follow the following order

$\text{keys}(T1) < \text{key}(x) < \text{keys}(T2) < \text{key}(y) < \text{keys}(T3)$

So BST property is not violated anywhere.

Procedure to delete a node from AVL tree:

Let w be the node to be deleted

1. Perform standard BST delete for w.
2. Starting from w, travel up and find the first unbalanced node. Let z be the first unbalanced node, y be the larger height child of z, and x be the larger height child of y. Note that the definitions of x and y are different from insertion here.
3. Re-balance the tree by performing appropriate rotations on the subtree rooted with z. There can be 4 possible cases that needs to be handled as x, y and z can be arranged in 4 ways. Following are the possible 4 arrangements:
 - i. y is left child of z and x is left child of y (Left Left Case)
 - ii. y is left child of z and x is right child of y (Left Right Case)
 - iii. y is right child of z and x is right child of y (Right Right Case)
 - iv. y is right child of z and x is left child of y (Right Left Case)

```
class TreeNode
{
    int val, height;
    TreeNode left, right;

    TreeNode(int d) {
        val = d;
        height = 1;
    }
}

class AVL_Tree {

    TreeNode leftRotate(TreeNode z)
    {
        # Write code here
    }
}
```



```

    }

    TreeNode rightRotate(TreeNode z)
    {
        # Write code here
    }

    TreeNode insert(TreeNode node, int key)
    {
        # Write code here
    }

```

11.3 Count Greater Nodes in AVL Tree

Given an AVL tree, calculate number of elements which are greater than given value in AVL tree.

Input: x = 5

Root of below AVL tree

```

    9
   /\
  1 10
 /\  \
0 5  11
/\  /\
-1 2 6

```

Output: 4

Explanation: There are 4 values which are greater than 5 in AVL tree which are 6, 9, 10 and 11.

```

class TreeNode {
    int key, height, desc;
    TreeNode left, right;

    TreeNode(int d) {
        key = d;
        height = 1;
        desc = 0;
    }
}

class AVL_Tree
{
    # Write code here
}

TreeNode insert(TreeNode node, int key)
{
    # Write code here
}

TreeNode minValueNode(TreeNode node)
{
    # Write code here
}

TreeNode deleteNode(TreeNode root, int key)
{
    # Write code here
}

void preOrder(TreeNode node)
{

```

```

        # Write code here
    }

public class Main
{
    # Write code here
}

```

11.4 Minimum Number of Nodes in an AVL Tree with given Height

Given the height of an AVL tree 'h', the task is to find the minimum number of nodes the tree can have.

Input: H = 0

Output: N = 1

Only '1' node is possible if the height of the tree is '0' which is the root node.

Input: H = 3

Output: N = 7

Recursive approach:

In an AVL tree, we have to maintain the height balance property, i.e. difference in the height of the left and the right subtrees cannot be other than -1, 0 or 1 for each node.

We will try to create a recurrence relation to find minimum number of nodes for a given height, $n(h)$.

- For height = 0, we can only have a single node in an AVL tree, i.e. $n(0) = 1$
- For height = 1, we can have a minimum of two nodes in an AVL tree, i.e. $n(1) = 2$
- Now for any height 'h', root will have two subtrees (left and right). Out of which one has to be of height h-1 and other of h-2. [root node excluded]
- So, $n(h) = 1 + n(h-1) + n(h-2)$ is the required recurrence relation for $h \geq 2$ [1 is added for the root node]

```

public class AVLTreeMinimumNodes {

    public static int AVLnodes(int height)
    {
        # Write code here
    }

    public static void main(String[] args) {
        int H = 3;
        System.out.println(AVLnodes(H)); // Output: 4
    }
}

```

12. Graph Traversal

12.1 Breadth First Search

The **Breadth First Search (BFS)** algorithm is used to search a graph data structure for a node that meets a set of criteria. It starts at the root of the graph and visits all nodes at the current depth level before moving on to the nodes at the next depth level.

For a given graph G, print BFS traversal from a given source vertex.

```

import java.util.*;

```

```

public class Graph {
    private Map<Integer, List<Integer>> graph;

    public Graph()
    {
        graph = new HashMap<>();
    }

    public void addEdge(int u, int v) {
        if (!graph.containsKey(u)) {
            graph.put(u, new ArrayList<>());
        }
        graph.get(u).add(v);
    }

    public void BFS(int s)
    {
        # Write code here
    }

    public static void main(String[] args)
    {
        # Write code here
    }
}

```

Output: Following is Breadth First Traversal (starting from vertex 2)

2 0 3 1

12.2 Depth First Search

Depth First Traversal (or DFS) for a graph is similar to Depth First Traversal of a tree. The only catch here is, that, unlike trees, graphs may contain cycles (a node may be visited twice). To avoid processing a node more than once, use a boolean visited array. A graph can have more than one DFS traversal.

For a given graph G, print DFS traversal from a given source vertex.

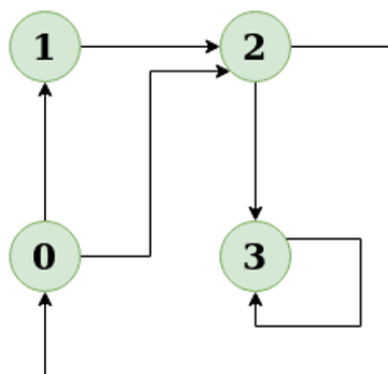
Input: n = 4, e = 6

0 -> 1, 0 -> 2, 1 -> 2, 2 -> 0, 2 -> 3, 3 -> 3

Output: DFS from vertex 1: 1 2 0 3

Explanation:

DFS Diagram:



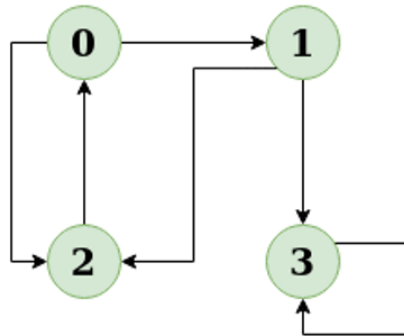
Input: n = 4, e = 6

2 -> 0, 0 -> 2, 1 -> 2, 0 -> 1, 3 -> 3, 1 -> 3

Output: DFS from vertex 2: 2 0 1 3

Explanation:

DFS Diagram:



```
import java.util.*;
class Graph {
    private Map<Integer, List<Integer>> graph;

    public Graph() {
        // Initialize the graph as a HashMap of ArrayLists
        graph = new HashMap<>();
    }
    public void addEdge(int u, int v)
    {
        # Write code here
    }

    public void DFS(int v) {
        DFSUtil(v, visited);
    }
    public static void main(String[] args)
    {
        # Write code here
    }
}
```

12.3 Best First Search (Informed Search)

The idea of Best First Search is to use an evaluation function to decide which adjacent is most promising and then explore. Best First Search falls under the category of Heuristic Search or Informed Search.

Implementation of Best First Search:

We use a priority queue or heap to store the costs of nodes that have the lowest evaluation function value. So the implementation is a variation of BFS, we just need to change Queue to PriorityQueue.

Algorithm:

Best-First-Search(Graph g, Node start)

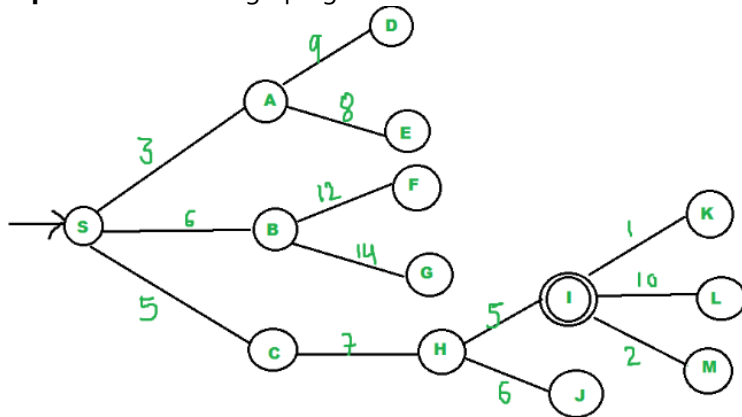
- 1) Create an empty PriorityQueue
PriorityQueue pq;
- 2) Insert "start" in pq.
pq.insert(start)
- 3) Until PriorityQueue is empty

```

u = PriorityQueue.DeleteMin
If u is the goal
    Exit
Else
    For each neighbor v of u
        If v "Unvisited"
            Mark v "Visited"
            pq.insert(v)
        Mark u "Examined"
End procedure

```

Input: Consider the graph given below.



- We start from source "S" and search for goal "I" using given costs and Best First search.
- pq initially contains S
- We remove S from pq and process unvisited neighbors of S to pq.
- pq now contains {A, C, B} (C is put before B because C has lesser cost)
- We remove A from pq and process unvisited neighbors of A to pq.
- pq now contains {C, B, E, D}
- We remove C from pq and process unvisited neighbors of C to pq.
- pq now contains {B, H, E, D}
- We remove B from pq and process unvisited neighbors of B to pq.
- pq now contains {H, E, D, F, G}
- We remove H from pq.
- Since our goal "I" is a neighbor of H, we return.

```

import java.util.*;

public class BestFirstSearch {
    static int v = 14;
    static List<List<Pair<Integer, Integer>>> graph = new ArrayList<>();
    static void addedge(int x, int y, int cost) {
        graph.get(x).add(new Pair<>(y, cost));
        graph.get(y).add(new Pair<>(x, cost));
    }

    static void best_first_search(int actual_Src, int target, int n)
    {
        # Write code here
    }
}

```

```

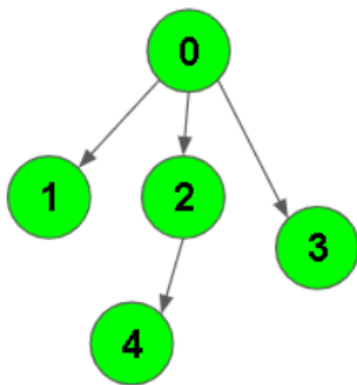
public static void main(String[] args)
{
    # Write code here
}

```

12.4 Breadth First Traversal of a Graph

Given a directed graph. The task is to do Breadth First Traversal of this graph starting from 0. One can move from node u to node v only if there's an edge from u to v . Find the BFS traversal of the graph starting from the 0th vertex, from left to right according to the input graph. Also, you should only take nodes directly or indirectly connected from Node 0 in consideration.

Input: Consider the graph given below where $V = 5$, $E = 4$, edges = $\{(0,1), (0,2), (0,3), (2,4)\}$



Output: 0 1 2 3 4

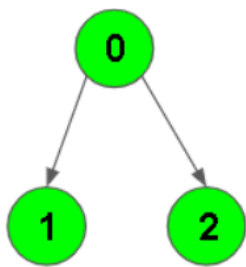
Explanation:

0 is connected to 1, 2, and 3.

2 is connected to 4.

So starting from 0, it will go to 1 then 2 then 3. After this 2 to 4, thus BFS will be 0 1 2 3 4.

Input: Consider the graph given below where $V = 3$, $E = 2$, edges = $\{(0, 1), (0, 2)\}$



Output: 0 1 2

Explanation:

0 is connected to 1, 2. So starting from 0, it will go to 1 then 2, thus BFS will be 0 1 2.

Your task is to complete the function **bfsOfGraph()** which takes the integer V denoting the number of vertices and adjacency list as input parameters and returns a list containing the BFS traversal of the graph starting from the 0th vertex from left to right.

```

import java.util.*;

class Graph {
    private int V;
    private LinkedList<Integer>[] adj;
}

```

```

Graph(int v) {
    V = v;
    adj = new LinkedList[v];
    for (int i = 0; i < v; ++i)
        adj[i] = new LinkedList();
}

void addEdge(int v, int w) {
    adj[v].add(w);
}

void BFS(int s)
{
    # Write Code Here
}

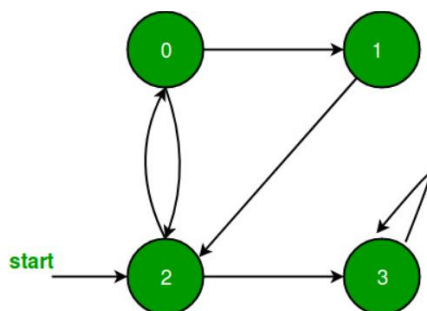
public static void main(String args[]) {
    # Write Code Here
}
}

```

12.5 Depth First Search (DFS) for Disconnected Graph

Given a Disconnected Graph, the task is to implement DFS or Depth First Search Algorithm for this Disconnected Graph.

Input: Consider the graph given below.



Output: 0 1 2 3

Procedure for DFS on Disconnected Graph:

Iterate over all the vertices of the graph and for any unvisited vertex, run a DFS from that vertex.

```

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

// Class representing a directed graph using adjacency list representation
class Graph
{
    # Write Code Here
}

public Graph()
{
    graph = new HashMap<>();
}

```

```

public void addEdge(int u, int v)
{
    # Write Code Here
}
private void DFSUtil(int v, boolean[] visited)
{
    # Write Code Here
}

public void DFS() {
    boolean[] visited = new boolean[graph.size()];
    # Write Code Here
}

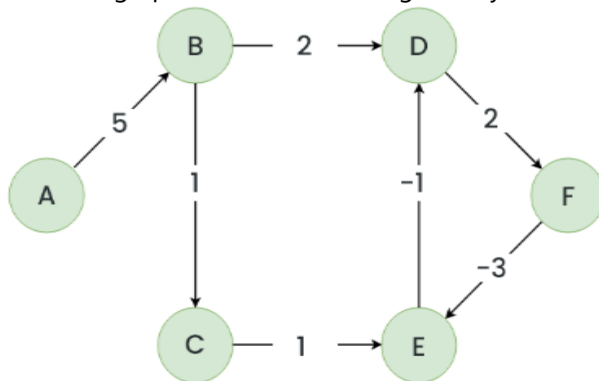
public static void main(String[] args)
{
    # Write Code Here
}
}

```

Try:

1. **Detect a negative cycle in a Graph (Bellman Ford):** A Bellman-Ford algorithm is also guaranteed to find the shortest path in a graph, similar to Dijkstra's algorithm. Although Bellman-Ford is slower than Dijkstra's algorithm, it is capable of handling graphs with negative edge weights, which makes it more versatile. The shortest path cannot be found if there exists a negative cycle in the graph. If we continue to go around the negative cycle an infinite number of times, then the cost of the path will continue to decrease (even though the length of the path is increasing).

Consider a graph G and detect a negative cycle in the graph using Bellman Ford algorithm.



13. Minimum Spanning Tree (MST)

13.1 Kruskal's Algorithm

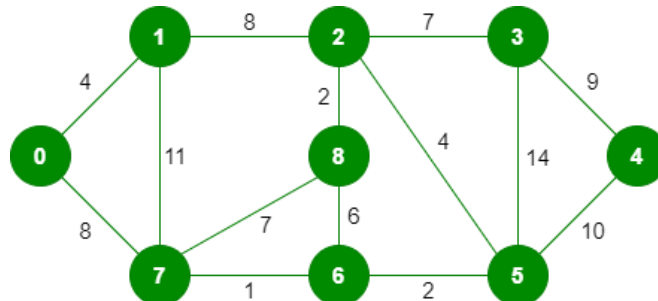
In Kruskal's algorithm, sort all edges of the given graph in increasing order. Then it keeps on adding new edges and nodes in the MST if the newly added edge does not form a cycle. It picks the minimum weighted edge at first and the maximum weighted edge at last.

MST using Kruskal's algorithm:

4. Sort all the edges in non-decreasing order of their weight.
5. Pick the smallest edge. Check if it forms a cycle with the spanning tree formed so far. If the cycle is not formed, include this edge. Else, discard it.
6. Repeat step#2 until there are (V-1) edges in the spanning tree.

Kruskal's algorithm to find the minimum cost spanning tree uses the greedy approach. The Greedy Choice is to pick the smallest weight edge that does not cause a cycle in the MST constructed so far.

Input: For the given graph G find the minimum cost spanning tree.



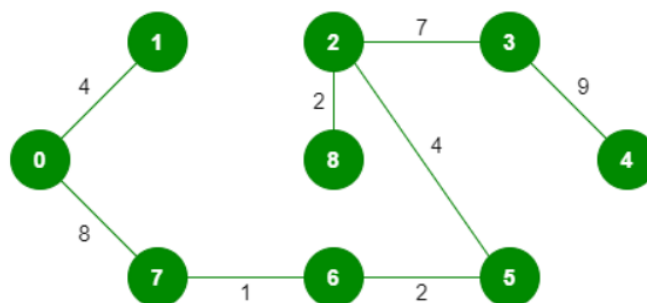
The graph contains 9 vertices and 14 edges. So, the minimum spanning tree formed will be having $(9 - 1) = 8$ edges.

After sorting:

Weight	Source	Destination
1	7	6
2	8	2
2	6	5
4	0	1
4	2	5
6	8	6
7	2	3
7	7	8
8	0	7
8	1	2
9	3	4
10	5	4
11	1	7
14	3	5

Now pick all edges one by one from the sorted list of edges.

Output:



```
// Kruskal's algorithm to find minimum Spanning Tree of a given connected,
import java.util.*;
```

```

public class Graph {

    class Edge implements Comparable<Edge> {
        int src, dest, weight;

        // Comparator function used for sorting edges
        public int compareTo(Edge compareEdge) {
            return this.weight - compareEdge.weight;
        }
    }

    private int V; // Number of vertices
    private List<Edge> edges; // List of edges

    public Graph(int vertices) {
        this.V = vertices;
        this.edges = new ArrayList<>();
    }

    public void addEdge(int u, int v, int w)
    {
        # Write Code Here
    }

    private void union(int[] parent, int[] rank, int x, int y)
    {
        # Write Code Here
    }

    public void KruskalMST()
    {
        # Write Code Here
    }

    public static void main(String[] args) {
        Graph g = new Graph(4);
        g.addEdge(0, 1, 10);
        g.addEdge(0, 2, 6);
        g.addEdge(0, 3, 5);
        g.addEdge(1, 3, 15);
        g.addEdge(2, 3, 4);

        // Function call
        g.KruskalMST();
    }
}

```

Output: Following are the edges in the constructed MST

2 -- 3 == 4

0 -- 3 == 5

0 -- 1 == 10

Minimum Cost Spanning Tree: 19

13.2 Prim's Algorithm

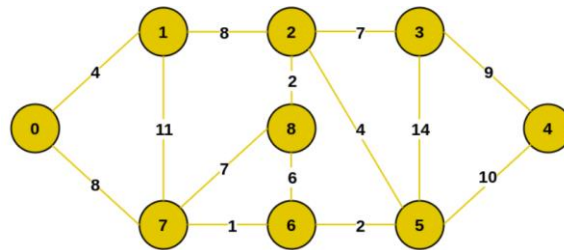
The Prim's algorithm starts with an empty spanning tree. The idea is to maintain two sets of vertices. The first set contains the vertices already included in the MST, and the other set contains the vertices not yet included. At every step, it considers all the edges that connect the two sets and picks the minimum weight edge from these edges. After picking the edge, it moves the other endpoint of the edge to the set containing MST.

Prim's Algorithm:

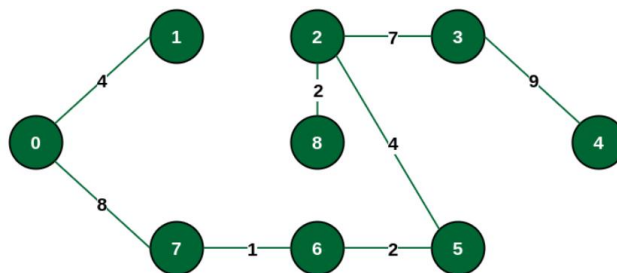
The working of Prim's algorithm can be described by using the following steps:

7. Determine an arbitrary vertex as the starting vertex of the MST.
8. Follow steps 3 to 5 till there are vertices that are not included in the MST (known as fringe vertex).
9. Find edges connecting any tree vertex with the fringe vertices.
10. Find the minimum among these edges.
11. Add the chosen edge to the MST if it does not form any cycle.
12. Return the MST and exit

Input: For the given graph G find the minimum cost spanning tree.



Output: The final structure of the MST is as follows and the weight of the edges of the MST is $(4 + 8 + 1 + 2 + 4 + 2 + 7 + 9) = 37$.



```
import java.util.Arrays;

public class Graph {

    private int V; // Number of vertices
    private int[][] graph; // Adjacency matrix representation of graph

    public Graph(int vertices) {
        this.V = vertices;
        this.graph = new int[V][V];
    }

    public void printMST(int[] parent) {
        System.out.println("Edge \tWeight");
        for (int i = 1; i < V; i++) {
            System.out.println(parent[i] + " - " + i + "\t" + graph[i][parent[i]]);
        }
    }

    private int minKey(int[] key, boolean[] mstSet)
    {
        # Write Code Here
    }

    public void primMST()
    {
        # Write Code Here
    }
}
```

```

    }
    public static void main(String[] args)
    {
        # Write Code Here
    }
}

```

Output:

Edge	Weight
0 - 1	2
1 - 2	3
0 - 3	6
1 - 4	5

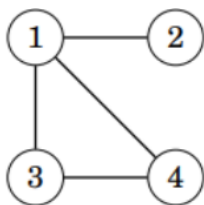
13.3 Total Number of Spanning Trees in a Graph

If a graph is a complete graph with n vertices, then total number of spanning trees is $n^{(n-2)}$ where n is the number of nodes in the graph. In complete graph, the task is equal to counting different labeled trees with n nodes for which have Cayley's formula.

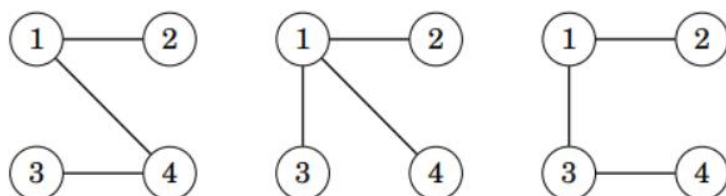
Laplacian matrix:

A Laplacian matrix L , where $L[i, i]$ is the degree of node i and $L[i, j] = -1$ if there is an edge between nodes i and j , and otherwise $L[i, j] = 0$.

Kirchhoff's theorem provides a way to calculate the number of spanning trees for a given graph as a determinant of a special matrix. Consider the following graph,



All possible spanning trees are as follows:



In order to calculate the number of spanning trees, construct a Laplacian matrix L , where $L[i, i]$ is the degree of node i and $L[i, j] = -1$ if there is an edge between nodes i and j , and otherwise $L[i, j] = 0$.

for the above graph, The Laplacian matrix will look like this

$$L = \begin{bmatrix} 3 & -1 & -1 & -1 \\ -1 & 1 & 0 & 0 \\ -1 & 0 & 2 & -1 \\ -1 & 0 & -1 & 2 \end{bmatrix}$$

The number of spanning trees equals the determinant of a matrix.

The Determinant of a matrix that can be obtained when we remove any row and any column from L .

For example, if we remove the first row and column, the result will be,

$$\det \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix} = 3.$$

The determinant is always the same, regardless of which row and column we remove from L.

```
import java.util.Arrays;
```

```
public class NumberOfSpanningTrees {

    static final int MAX = 100;
    static final int MOD = 1000000007;
    void multiply(long[][] A, long[][] B, long[][] C, int size) {
        for (int i = 0; i < size; i++) {
            for (int j = 0; j < size; j++) {
                C[i][j] = 0;
                for (int k = 0; k < size; k++) {
                    C[i][j] = (C[i][j] + A[i][k] * B[k][j]) % MOD;
                }
            }
        }
    }
    void power(long[][] A, int N, long[][] result, int size)
    {
        # Write Code Here
    }

    long numOfSpanningTree(int[][] graph, int V)
    {
        # Write Code Here
    }

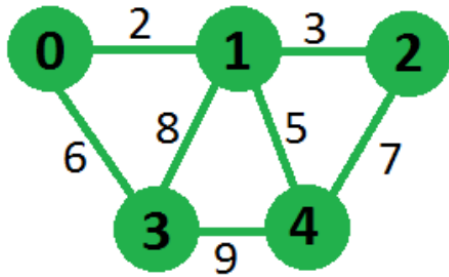
    public static void main(String[] args) {
        int V = 4; // Number of vertices in graph
        int E = 5; // Number of edges in graph
        int[][] graph = { { 0, 1, 1, 1 }, { 1, 0, 1, 1 }, { 1, 1, 0, 1 }, { 1, 1, 1,
0 } };

        NumberOfSpanningTrees obj = new NumberOfSpanningTrees();
        System.out.println(obj.numOfSpanningTree(graph, V));
    }
}
```

13.4 Minimum Product Spanning Tree

A minimum product spanning tree for a weighted, connected, and undirected graph is a spanning tree with a weight product less than or equal to the weight product of every other spanning tree. The weight product of a spanning tree is the product of weights corresponding to each edge of the spanning tree. All weights of the given graph will be positive for simplicity.

Input:



Output: Minimum Product that we can obtain is 180 for above graph by choosing edges 0-1, 1-2, 0-3 and 1-4

This problem can be solved using standard minimum spanning tree algorithms like Kruskal and prim's algorithm, but we need to modify our graph to use these algorithms. Minimum spanning tree algorithms tries to minimize the total sum of weights, here we need to minimize the total product of weights. We can use the property of logarithms to overcome this problem.

$$\log(w_1 * w_2 * w_3 * \dots * w_N) = \log(w_1) + \log(w_2) + \log(w_3) \dots + \log(w_N)$$

We can replace each weight of the graph by its log value, then we apply any minimum spanning tree algorithm which will try to minimize the sum of $\log(w_i)$ which in turn minimizes the weight product.

```
import java.util.Arrays;
```

```
public class MinimumProductMST {
    // Number of vertices in the graph
    static final int V = 5;

    // A utility function to find the vertex with minimum key value, from the set of
    // vertices not yet included in MST
    int minKey(int key[], boolean mstSet[]) {
        int min = Integer.MAX_VALUE, min_index = -1;

        for (int v = 0; v < V; v++) {
            if (mstSet[v] == false && key[v] < min) {
                min = key[v];
                min_index = v;
            }
        }

        return min_index;
    }

    void printMST(int parent[], int n, int graph[][]){
        {
            # Write Code Here
        }
    }
    void primMST(int inputGraph[][], int logGraph[][]){
        {
            # Write Code Here
        }
    }
    void minimumProductMST(int graph[][]){
        {
            # Write Code Here
        }
    }
    public static void main(String[] args)
    {
        # Write Code Here
    }
}
```

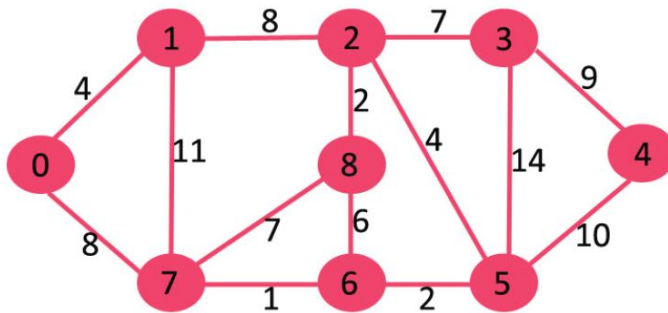
13.5 Reverse Delete Algorithm for Minimum Spanning Tree

In Reverse Delete algorithm, we sort all edges in decreasing order of their weights. After sorting, we one by one pick edges in decreasing order. We include current picked edge if excluding current edge causes disconnection in current graph. The main idea is delete edge if its deletion does not lead to disconnection of graph.

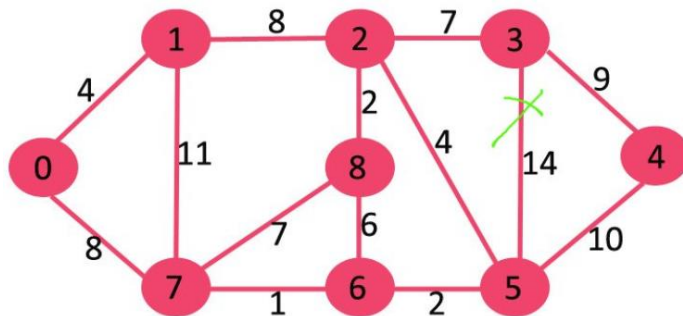
Algorithm:

1. Sort all edges of graph in non-increasing order of edge weights.
2. Initialize MST as original graph and remove extra edges using step 3.
3. Pick highest weight edge from remaining edges and check if deleting the edge disconnects the graph or not.
If disconnects, then we don't delete the edge.
Else we delete the edge and continue.

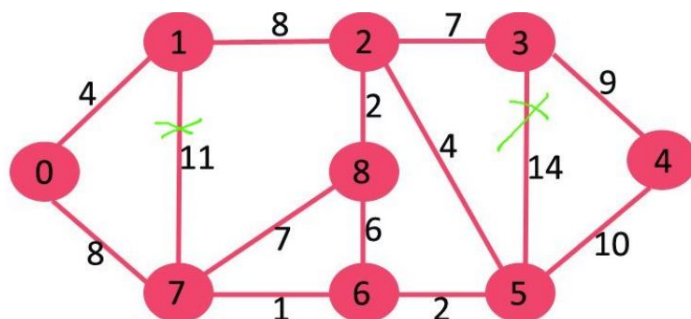
Input: Consider the graph below



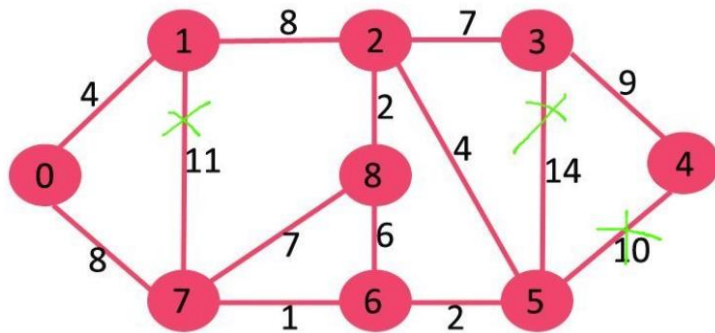
If we delete highest weight edge of weight 14, graph doesn't become disconnected, so we remove it.



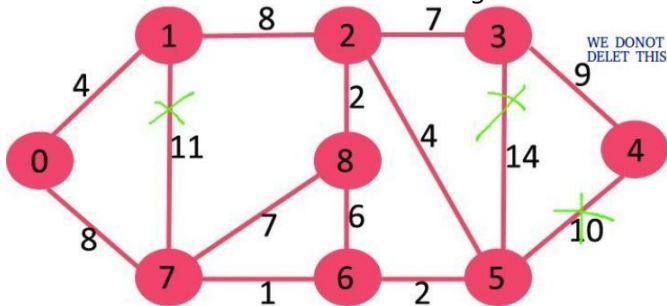
Next we delete 11 as deleting it doesn't disconnect the graph.



Next we delete 10 as deleting it doesn't disconnect the graph.



Next is 9. We cannot delete 9 as deleting it causes disconnection.



We continue this way and following edges remain in final MST.

Edges in MST

(3, 4)
(0, 7)
(2, 3)
(2, 5)
(0, 1)
(5, 6)
(2, 8)
(6, 7)

```
import java.util.ArrayList;
import java.util.Collections;

// Edge class to represent edges in the graph
class Edge {
    int src, dest, weight;

    Edge(int src, int dest, int weight) {
        this.src = src;
        this.dest = dest;
        this.weight = weight;
    }
}

class Graph
{
    # Write Code Here
}

// Function to add an edge to the graph
void addEdge(int u, int v, int w) {
    # Write Code Here
}
```



```

void dfs(int v, boolean[] visited)
{
    # Write Code Here
}

// Function to check if the graph is connected
boolean connected()
{
    # Write Code Here
}

void reverseDeleteMST()
{
    # Write Code Here
}

}

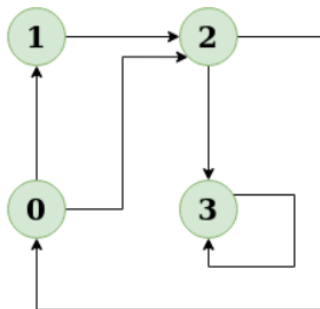
public class ReverseDeleteMST {
    public static void main(String[] args)
    {
        # Write Code Here
    }
}

```

Try:

1. **Detect Cycle in a Directed Graph:** Given the root of a Directed graph, The task is to check whether the graph contains a cycle or not.

Input: N = 4, E = 6



Output: Yes

Explanation: The diagram clearly shows a cycle 0 -> 2 -> 0

14. Final Notes

The only way to learn programming is program, program and program on challenging problems. The problems in this tutorial are certainly NOT challenging. There are tens of thousands of challenging problems available – used in training for various programming contests (such as International Collegiate Programming Contest (ICPC), International Olympiad in Informatics (IOI)). Check out these sites:

- The ACM - ICPC International collegiate programming contest (<https://icpc.global/>)
- The Topcoder Open (TCO) annual programming and design contest (<https://www.topcoder.com/>)
- Universidad de Valladolid's online judge (<https://uva.onlinejudge.org/>).
- Peking University's online judge (<http://poj.org/>).
- USA Computing Olympiad (USACO) Training Program @ <http://train.usaco.org/usacogate>.
- Google's coding competitions (<https://codingcompetitions.withgoogle.com/codejam>,
<https://codingcompetitions.withgoogle.com/hashcode>)
- The ICFP programming contest (<https://www.icfpconference.org/>)
- BME International 24-hours programming contest (<https://www.challenge24.org/>)
- The International Obfuscated C Code Contest (<https://www0.us.ioccc.org/main.html>)
- Internet Problem Solving Contest (<https://ipsc.ksp.sk/>)
- Microsoft Imagine Cup (<https://imaginecup.microsoft.com/en-us>)
- Hewlett Packard Enterprise (HPE) Codewars (<https://hpecodewars.org/>)
- OpenChallenge (<https://www.openchallenge.org/>)

Coding Contests Scores

Students must solve problems and attain scores in the following coding contests:

Name of the contest	Minimum number of problems to solve	Required score
• CodeChef	20	200
• Leetcode	20	200
• GeeksforGeeks	20	200
• SPOJ	5	50
• InterviewBit	10	1000
• Hackerrank	25	250
• Codeforces	10	100
• BuildIT	50	500
Total score need to obtain		2500

Student must have any one of the following certification:

2. HackerRank - Problem Solving Skills Certification (Basic and Intermediate)
2. GeeksforGeeks – Data Structures and Algorithms Certification
3. CodeChef - Learn Data Structures and Algorithms Certification
4. Interviewbit – DSA pro / Python pro
5. Edx – Data Structures and Algorithms
5. NPTEL – Programming, Data Structures and Algorithms
6. NPTEL – Introduction to Data Structures and Algorithms
7. NPTEL – Data Structures and Algorithms
8. NPTEL – Programming and Data Structure

V. TEXT BOOKS:

1. Rance D. Necaise, "Data Structures and Algorithms using Python", Wiley Student Edition.
2. Benjamin Baka, David Julian, "Python Data Structures and Algorithms", Packt Publishers, 2017.

VI. REFERENCE BOOKS:

1. S. Lipschutz, "Data Structures", Tata McGraw Hill Education, 1st Edition, 2008.
2. D. Samanta, "Classic Data Structures", PHI Learning, 2nd Edition, 2004.

VII. ELECTRONICS RESOURCES:

1. https://www.tutorialspoint.com/data_structures_algorithms/algorithms_basics.htm
2. <https://www.codechef.com/certification/data-structures-and-algorithms/prepare>
3. <https://www.cs.auckland.ac.nz/software/AlgAnim/dsToC.html>
4. <https://online-learning.harvard.edu/course/data-structures-and-algorithms>

VIII. MATERIALS ONLINE

1. Course Template
2. Lab manual



INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal - 500 043, Hyderabad, Telangana

EXPERIENTIAL ENGINEERING EDUCATION (EXEED) – PROTOTYPE / DESIGN BUILDING

III Semester: AE / ME / CE / ECE / EEE / CSE / CSE (AI & ML) / CSE (DS) / CSE (CS) / IT								
Course Code	Category	Hours / Week			Credits	Maximum Marks		
ACSD12	Core	L	T	P	C	CIA	SEE	Total
		0	0	2	1	40	60	100
Contact Classes: NIL	Tutorial Classes: NIL	Practical Classes:45				Total Classes: 45		
Prerequisite: Essentials of Innovation.								

I. COURSE OVERVIEW:

This course provides an overall exposure to the various methods and tools of prototyping. This course discusses Low- Fidelity, paper, wireframing and tool-based prototyping techniques along with design principles and patterns.

II. COURSES OBJECTIVES:

The students will try to learn:

- I. The basic principles and design aspect of prototyping.
- II. The various techniques, design guidelines and patterns.
- III. The applications of prototyping using various tools and platforms.

III. COURSE CONTENT:

Module 1: An Introduction to Prototyping

Objective: Understand the basics of prototyping and its importance in the design process.

Exercise:

1. Introduction Lecture (10 minutes): Overview of prototyping and its types.
2. Brainstorming Session (20 minutes): Brainstorm simple app ideas.
3. Sketching Session (20 minutes): Create initial design sketches on paper.
4. Paper Prototyping (40 minutes): Build a low-fidelity paper prototype of the app's user interface.
5. Presentation and Feedback (20 minutes): Present prototypes and receive feedback.

Materials and Tools: Paper, pencils, rulers, scissors, glue.

Module 2: Low-Fidelity Prototyping and Paper Prototyping

Objective: Learn concepts and techniques of low-fidelity and paper prototyping.

Exercise:

1. Lecture (10 minutes): Concepts of low-fidelity prototyping.
2. Hands-On Session (40 minutes): Build paper prototypes of the app's screens and user flow.
3. Peer Review (30 minutes): Exchange prototypes with peers for feedback.
4. Iteration (30 minutes): Refine prototypes based on feedback.

Materials and Tools: Paper, pencils, rulers, scissors, glue.

Module 3: Wireframing and Tool-Based Prototyping

Objective: Create wireframes and digital prototypes using software tools.

Exercise:

1. Lecture (10 minutes): Introduction to wireframing and digital prototyping tools.
2. Wireframing (30 minutes): Design wireframes of the app's user interface using tools like Figma or Sketch.
3. Tool-Based Prototyping (50 minutes): Create a digital prototype of the app.
4. Presentation and Feedback (20 minutes): Share digital prototypes and gather feedback.

Materials and Tools: Wireframing tools, computers.

Module 4: Physical Low-Fidelity Prototyping and 3D Printing

Objective: Build simple physical prototypes using basic materials and introduce 3D printing.

Exercise:

1. Lecture (10 minutes): Importance of physical prototyping.
2. Prototype Building (30 minutes): Create a physical representation of the app's main interface using materials like cardboard and foam.
3. **Introduction to 3D Printing (20 minutes):** Overview of 3D printing technology and its applications in prototyping.
4. Testing (20 minutes): Test the physical prototypes for basic functionality.
5. Feedback Session (30 minutes): Present and discuss prototypes.

Materials and Tools: Cardboard, foam, clay, scissors, glue, 3D printer.

Module 5: Tool-Based Prototyping and Circuit Design

Objective: Develop advanced digital prototypes using specialized tools and introduce basic circuit design concepts.

Exercise:

1. Lecture (10 minutes): Advanced prototyping tools and techniques, introduction to circuit design.
2. Hands-On Session (40 minutes): Use tools like Figma, Adobe XD, or Android Studio to create an interactive prototype of the app.
3. **Circuit Design (20 minutes):** Design simple circuits that could be integrated into the app's hardware prototype.
4. User Testing (20 minutes): Conduct user tests and document feedback.
5. Feedback and Iteration (20 minutes): Refine prototypes based on user feedback.

Materials and Tools: Advanced prototyping software, computers, circuit design tools (e.g., breadboards, resistors, LEDs).

Module 6: Design Principles and Patterns - Graphic Design

Objective: Apply graphic design principles to create visually appealing app interfaces.

Exercise:

1. Lecture (10 minutes): Graphic design principles (contrast, alignment, repetition, proximity).
2. Design Session (50 minutes): Create a graphic design layout for the app's interface.
3. Peer Review (30 minutes): Exchange designs for feedback.
4. Refinement (20 minutes): Refine designs based on peer feedback.

Materials and Tools: Graphic design software, computers.

Module 7: Interaction Design and Arduino Integration

Objective: Apply interaction design principles to create intuitive app interfaces and introduce Arduino for hardware interactions.

Exercise:

1. Lecture (10 minutes): Basics of interaction design and Arduino integration.
2. Design Session (40 minutes): Develop interactive wireframes or prototypes for the app's user interactions.
3. **Arduino Integration (20 minutes):** Connect simple Arduino circuits to the app prototype to simulate interactions (e.g., button presses, LED responses).
4. User Testing (20 minutes): Conduct usability tests with peers.
5. Iteration (20 minutes): Improve designs based on test results.

Materials and Tools: Interaction design tools, computers, Arduino kits.

Module 8: Commercial Design Guidelines and Standards

Objective: Design according to industry standards and guidelines.

Exercise:

1. Lecture (10 minutes): Overview of commercial design guidelines
2. Design Session (50 minutes): Develop a prototype of the app following specific design guidelines.
3. Review (30 minutes): Evaluate prototypes against the guidelines.
4. Presentation and Discussion (20 minutes): Present prototypes and discuss adherence to guidelines.

Materials and Tools: Design guidelines documents, design software, computers.

Module 9: Universal Design: Sensory and Cognitive Impairments

Objective: Create app designs accessible to users with sensory and cognitive impairments.

Exercise:

1. Lecture (10 minutes): Principles of universal design.
2. Design Session (50 minutes): Develop an accessible design for the app.

3. User Testing (20 minutes): Test designs with peers simulating impairments.
4. Feedback Session (30 minutes): Present and discuss accessibility features.

Materials and Tools: Design software, computers, accessibility guidelines.

Module 10: Universal Design: Tools, Limitations, and Standards

Objective: Understand tools and limitations in universal design, and adhere to standards.

Exercise:

1. Lecture (10 minutes): Tools and limitations in universal design.
2. Hands-On Session (50 minutes): Use tools to create universally designed app prototypes.
3. Testing (20 minutes): Evaluate prototypes for accessibility and adherence to standards.
4. Iteration (30 minutes): Refine designs based on testing outcomes.

Materials and Tools: Universal design tools, computers.

Module 11: Mobile UI Design, CNC, and Arduino

Objective: Design user interfaces for mobile and wearable devices, create related hardware prototypes using CNC tools, and integrate Arduino for interactions.

Exercise:

1. Lecture (10 minutes): Principles of mobile UI and wearable design, introduction to CNC lathe and mill, and Arduino for wearable interactions.
2. Design Session (30 minutes): Develop a UI prototype for a mobile app or wearable device.
3. **CNC Prototyping (20 minutes):** Create physical components of the wearable device using CNC lathe and mill.
4. **Arduino Integration (20 minutes):** Program Arduino for basic interactions (e.g., motion detection, feedback via LEDs).
5. User Testing (20 minutes): Conduct usability tests on the prototype.
6. Feedback Session (30 minutes): Present and discuss usability findings.

Materials and Tools: Mobile UI design tools, computers, mobile devices, CNC lathe, CNC mill, Arduino kits.

Module 12: Automotive User Interface, Laser Engraving, and Arduino Integration

Objective: Design user interfaces for automotive applications, integrate laser engraving for UI elements, and use Arduino for sensor-based interactions.

Exercise:

1. Lecture (10 minutes): Fundamentals of automotive UI design, overview of laser engraving, and Arduino for sensor integration.
2. Design Session (30 minutes): Create a prototype for an automotive interface (e.g., dashboard layout).

3. **Laser Engraving (20 minutes):** Use laser engraving to create high-precision UI elements for the automotive interface.
4. **Arduino Integration (20 minutes):** Implement Arduino to simulate sensor-based interactions (e.g., temperature control, obstacle detection).
5. **Testing (20 minutes):** Test the prototype for usability and functionality.
6. **Feedback Session (30 minutes):** Present and discuss design improvements.

Materials and Tools: Automotive UI design tools, computers, laser engraving machine, Arduino kits.

IV. REFERENCE BOOKS:

1. Chee Kai Chua, Kah Fai Leong, "3D Printing and Additive Manufacturing: Principles and Applications".
2. Simon Knight, "Arduino for Beginners: Step-by-Step Guide to Arduino (Arduino Hardware & Software)".
3. Peter Smid, "CNC Programming Handbook".
4. Steven Wolfe, "Laser Cutting and Engraving".
5. Charles Platt, "Make: Electronics: Learning Through Discovery".



INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal - 500 043, Hyderabad, Telangana

COURSE CONTENT

AC MACHINES								
IV Semester: EEE								
Course Code	Category	Hours / Week			Credits	Maximum Marks		
AEED10	Core	L	T	P	C	CIA	SEE	Total
		3	0	0	3	40	60	100
Contact Classes: 48	Tutorial Classes: Nil	Practical Classes: Nil			Total Classes: 48			

I. COURSE OVERVIEW:

This course deals with the basic theory, construction, operation, performance characteristics and application of electromechanical energy conversion devices such as synchronous and asynchronous machines. It also facilitates the study of the alternating machines which are the major part of industrial drives and agricultural pump sets.

II. COURSE OBJECTIVES:

The students will try to learn:

- I. The principle of operation and the effect of pulsating, rotating magnetic fields on the working of AC machines
- II. The armature winding layouts and concept of armature reaction with phasor diagrams.
- III. The starting, speed control methods and equivalent circuit diagram of poly phase and single phase machines.

III. COURSE OUTCOMES:

After successful completion of the course, students should be able to:

- CO1 **Demonstrate** various winding factors, spatially displaced armature windings to generate electro motive force in AC machines.
- CO2 **Illustrate** electromagnetic laws used for the construction and operation of synchronous and asynchronous machines.
- CO3 **Identify** various control strategies for calculating the performance parameters and voltage regulation of AC machines.
- CO4 **Demonstrate** the parallel operation of alternators for load sharing under various loading conditions.
- CO5 **Examine** the behavior of synchronous motor with variable excitation and loadings for calculating armature current, power and power factor

III. COURSE CONTENT:

MODULE-I: THREE PHASE INDUCTION MACHINES (10)

Magnetic fields: Constant magnetic field, pulsating magnetic field, rotating magnetic field; Three phase induction motors: Construction, types of induction motors, slip and frequency of rotor currents, rotor MMF and production of torque, equivalent circuit, power across air gap, torque and power output, torque slip characteristics, generating and braking modes, maximum (breakdown) torque, starting torque, maximum power output. Equivalent circuit; Induction generator: Operation, approximate equivalent circuit, doubly fed induction generator, numerical problems.

MODULE-II: TESTING AND SPEED CONTROL OF INDUCTION MOTORS (09)

Testing: Brake test, no load and blocked rotor test, determination of induction motor parameters from circle diagram, numerical problems. Starting methods of Slip ring and squirrel cage induction motors; Speed control of induction motors, numerical problems.

MODULE-III: ALTERNATORS (10)

Synchronous generators: Introduction, principle of operation, types, constructional features, integral slot and fractional slot windings, distributed and concentrated windings, winding factors, basic synchronous machine model, circuit model of a synchronous machine, armature reaction, phasor diagrams.



INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal - 500 043, Hyderabad, Telangana

MODULE-IV: SYNCHRONOUS MOTORS (10)

Synchronous motors: Principle of operation, phasor diagrams, power developed, synchronous motor with different excitations, effect of increased load with constant excitation, effect of change in excitation with constant load, effect of excitation on armature current and power factor, construction of “V” and inverted “V” curves, power and excitation circles, starting methods, and analysis, synchronous condenser.

MODULE-V: SINGLE-PHASE INDUCTION MOTORS (09)

Single phase induction motor: Principle of operation, two reaction theory, equivalent circuit analysis, split phase motor, construction, principle of operation, capacitor start, capacitor run, capacitor start - capacitor run motor, shaded pole motor, torque speed characteristics of single phase induction motors.

V. TEXT BOOKS:

1. P S Bimbhra, “Electrical Machinery”, Khanna Publishers, 1st edition, 2011.
2. I J Nagrath and D P Kothari, “Electric Machines”, McGraw Hill Education, 1st edition, 2010.
3. J B Gupta “Theory and performance of Electrical machines”, S.K.Kataria & Sons Publishers 14th edition, 2009.

VI. REFERENCE BOOKS:

1. M G Say, E O Taylor, “Direct Current Machines”, Longman Higher Education, 1st edition, 1985.
2. M G Say, “Performance and design of AC machines”, CBS Publishers, 1st edition, 2002.
3. A E Fitzgerald and C Kingsley, “Electric Machinery”, New York, McGraw Hill Education, 1st edition, 2013.
4. M V Deshpande, “Electrical Machines”, PHI Learning Private Limited, 3rd edition, 2011.

VII. ELECTRONIC RESOURCES:

1. <https://www.electrical4u.com>
2. <https://www.freevideolectures.com>

VIII. MATERIALS ONLINE:

1. Course template
2. Tutorial question bank
3. Tech talk topics
4. Open end experiments
5. Definitions and terminology
6. Assignments
7. Model question paper - I
8. Model question paper - II
9. Lecture notes
10. E-learning readiness videos (ELRV)
11. Power point presentation

COURSE CONTENT

POWER GENERATION SYSTEMS AND ECONOMIC ASPECTS								
III Semester: EEE								
Course Code	Category	Hours / Week			Credits	Maximum Marks		
AEED11	Core	L	T	P	C	CIA	SEE	Total
		3	0	0	3	40	60	100
Contact Classes: 48	Tutorial Classes: Nil	Practical Classes: Nil			Total Classes: 48			
Prerequisite: Nil								

I. COURSE OVERVIEW:

This course provides ability to recognize, analyze and troubleshoot different elements in electric power generation systems. It deals with conventional energy systems like thermal and nuclear power stations. This course also introduces non-conventional energy systems like solar energy (radiation, collection, storage, and application), Hydro and Wind energy. This course will also discuss some environmental impacts of power generation and also look at alternative and sustainable energy resources.

II. COURSE OBJECTIVES:

The students will try to learn:

- The fundamental concepts of power generation and gain knowledge about the different renewable and non-renewable energy sources.
- Thorough theory on the construction and working principle of thermal, hydro-electric, nuclear and gas power plants.
- The key aspects in solar and wind power energy systems and analyze their environmental aspects in the present-day scenario to obtain clean energy.
- The various factors affecting cost of generations and the different Tariff methods for electrical energy consumption to attain optimum utilization of generated electrical energy.
- The ability to incorporate the knowledge of electrical power generation in working with minor and major projects and to take up research work in future.

III. COURSE OUTCOMES:

After successful completion of the course, students should be able to:

- CO1 Demonstrate the knowledge about the electric power generations and their impacts.
- CO2 Assess the theory and practices of conventional and non-conventional power generation generation method.
- CO3 Determine the operation, maintenance and working of power plants.
- CO4 Aware of the principle of operation, components, layout, location, environmental and social issues of nuclear, diesel and gas power plant.
- CO5 Interpret the effect of role of tariff on the cost of power generation.

IV. COURSE CONTENT:

MODULE-I: Sources of Electrical Power (09)

Wind, solar, fuel cell, tidal, geo-thermal, hydro-electric, thermal-steam, diesel, gas, nuclear power plants (block diagram approach only). Concept of co-generation. Combined heat and power distributed generation. Diesel electric plants. Gas turbine plants. Mini, micro, and bio generation. Concept of distributed generation.

MODULE-II: Hydro Power Generation Thermal Power Generation (09)

Hydro Power Generation: Selection of site. Classification of hydro-electric plants. General arrangement and operation. Hydroelectric plant power station structure and control. Thermal Power Generation: Introduction. Main parts of a thermal power plant. Working. Plant layout. Pros and cons of nuclear power generation. Selection of site, cost, components of reactors. Description of fuel sources. Safety of nuclear power reactor.

Module-III: Solar Energy (09)

Solar radiation: Environmental impact of solar power, physics of the sun, solar constant, extraterrestrial and terrestrial solar radiation, solar radiation on tilted surface, instruments for measuring solar radiation, solar radiation data, solar concentrators, collectors, thermal applications, design of standalone solar systems, simple problems.

Photovoltaic systems: Photovoltaic effect, semiconducting materials, band gap theory, photo emission of

electrons, cell configuration, types of solar cells, cell properties, device physics, electrostatic field across the depletion layer, voltage developed, I-V characteristics, module structure and fabrication, output power and efficiency, fill factor, maximum power point tracking (MPPT), solar grid connected inverters, simple problems.

Module-IV: Wind Energy (09)

Wind energy: Sources and potential, power from wind, Betz criterion, components of wind energy conversion system, types of turbines, horizontal and vertical axis wind turbines, aerodynamics, momentum theory (actuator disk concept), operational characteristics, blade element theory, types of generating systems for wind energy.

Module-V: Economic Aspects of Power Generation (09)

Economics Aspects: Introduction. Terms commonly used in system operation. Diversity factor, load factor, plant capacity factor, plant use factor, plant utilization factor and loss factor, load duration curve. Cost of generating station, factors influencing the rate of tariff designing, tariff, types of tariff. Power factor improvement.

V. TEXT BOOKS:

1. C L Wadhawa, "Generation, Distribution and Utilization of Electrical Energy", New Age International Limited, New Delhi, 3rd edition, 2005.
2. G D Rai, "Non-Conventional Energy Sources", Khanna Publishers, 1st edition, 2011.
3. G N Tiwari, M K Ghosal, "Fundamentals of Renewable Energy Sources", Narosa Publications, New Delhi, 1st edition, 2007.
4. Chetan Singh Solanki, "Solar Photovoltaics", PHI Publications, 2nd edition, 2011.
5. M L Soni, P V Gupta, U S Bhatnagar and A Chakraborti, "A text book on Power system engineering", Dhanpat Rai and Co. Pvt. Ltd, 1999.

VI. REFERENCE BOOKS:

1. J B Gupta, "A Course in Electrical Power", S K Kataria and Sons, New Delhi, 15th edition, 2013.
2. M V Deshpande, "Elements of Power Station design", Prentice Hall India Learning Private Limited, New Delhi, 1st edition, 1992.
3. Mukund R Patel, "Wind and Solar Power Systems", CRC Press, 1st edition, 1999.
4. V K Mehta and Rohit Mehta, "Principle of Power Systems", S Chand & Company, Ltd, New Delhi, 3rd edition, 2005.

VII. ELECTRONICS RESOURCES:

1. <https://www.solarpowernotes.com>
2. <https://www.electrical4u.com/power-plants-types-of-power-plant>
3. <https://www.iare.ac.in>

VIII. MATERIALS ONLINE:

1. Course template
2. Tutorial question bank
3. Tech talk topics
4. Open end experiments
5. Definitions and terminology
6. Assignments
7. Model question paper - I
8. Model question paper - II
9. Lecture notes
10. E-learning readiness videos (ELRV)
11. Power point presentation



INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal - 500 043, Hyderabad, Telangana

COURSE CONTENT

CONTROL SYSTEMS								
IV Semester: EEE								
Course Code	Category	Hours / Week			Credits	Maximum Marks		
AEED12	Core	L	T	P	C	CIA	SEE	Total
		3	0	0	3	30	70	100
Contact Classes: 48	Tutorial Classes: Nil	Practical Classes: Nil			Total Classes: 48			
Prerequisite: Linear Algebra and Calculus , DC Machines and Transformers								

I. COURSE OVERVIEW:

This course deals with the basic concepts of block diagram reduction technique, time response analysis of first order and second order systems. It deals with various time and frequency domain analysis. It elaborates the concept of stability and its assessment for linear time invariant systems. This course address the various real time issues and how the control strategies are used in automation areas associates with variety of engineering streams

II. COURSE OBJECTIVES:

The students will try to learn:

- I. Organize modeling and analysis of electrical and mechanical systems.
- II. Analyze control systems by block diagrams and signal flow graph technique.
- III. Demonstrate the analytical and graphical techniques to study the stability.
- IV. Illustrate the frequency domain and state space analysis.

III. COURSE OUTCOMES:

The students will try to learn:

1. Relate the physical and mechanical systems into equivalent electrical analogies using the mathematical form of physical systems
2. Utilize various reduction techniques for developing the transfer function, transient and steady state error with the standard input signals
3. Make use of the ROUTH-HOURITZ criterion to determine the stability of a system
4. Demonstrate the stability of a system using root locus technique for analysing the system performance.
5. Illustrate the system using polar plot, Nyquist plot, and Bode plot for determining the stability of the system.
6. Interpret linear system equations in state space form for the analysis of LTI system.

IV. COURSE CONTENT:

MODULE-I: INTRODUCTION AND MODELING OF PHYSICAL SYSTEMS (08)

Control systems: Introduction, open loop and closed loop systems, examples, comparison, mathematical modeling and differential equations of physical systems, concept of transfer function, translational and rotational mechanical systems, electrical systems, force - voltage and force - current analogy.

MODULE-II: BLOCK DIAGRAM REDUCTION, TIME RESPONSE ANALYSIS AND CONTROLLERS (10)

Block Diagrams: Block diagram representation of various systems, block diagram algebra, characteristics of feedback systems, AC servomotor, transfer function of DC motor, signal flow graph, Mason's gain formula; Time response analysis: Standard test signals, shifted unit step, impulse response, unit step response of first and second order systems, time response specifications, steady state errors and error constants, dynamic error coefficients method, effects of P, PD, PI and PID controllers. Programmable Logic Controller: Physical structure of PLC, Types of PLCs, PLC applications, PLC programming and examples

MODULE-III: CONCEPT OF STABILITY AND ROOT LOCUS TECHNIQUE (09)

Concept of stability: Necessary and sufficient conditions for stability, Routh's and Routh Hurwitz stability criteria and limitations.

Root locus technique: Introduction, root locus concept, construction of root loci, graphical determination of 'k' for

specified damping ratio, relative stability, effect of adding zeros and poles on stability.

MODULE-IV: FREQUENCY DOMAIN ANALYSIS (10)

Frequency domain analysis: Introduction, frequency domain specifications, stability analysis from Bode plot, Nyquist plot, calculation of gain margin and phase margin, determination of transfer function, correlation between time and frequency responses.

MODULE-V: STATE SPACE ANALYSIS AND COMPENSATORS (08)

State Space Analysis: Concept of state, state variables and state model, derivation of state models from block diagrams, diagonalization, solving the time invariant state equations, state transition matrix and properties, concept of controllability and observability; Compensators: Lag, lead, lead - lag networks.

V. TEXT BOOKS:

1. I J Nagrath, M Gopal, "Control Systems Engineering", New Age International Publications, 3rd edition, 2007.
2. K Ogata, "Modern Control Engineering", Prentice Hall, 4th edition, 2003.
3. N C Jagan, "Control Systems", BS Publications, 1st edition, 2007.

VI. REFERENCE BOOKS:

1. Anand Kumar, "Control Systems", PHI Learning, 1st edition, 2007.
2. S Palani, "Control Systems Engineering", Tata McGraw-Hill Publications, 1st edition, 2001.
3. N K Sinha, "Control Systems", New Age International Publishers, 1st edition, 2002.

VII. ELECTRONIC RESOURCES:

1. <https://www.researchgate.net>
2. <https://www.aar.faculty.asu.edu/classes>
3. <https://www.facstaff.bucknell.edu/>
4. <https://www.electrical4u.com>
5. <https://www.iare.ac.in>

VIII. MATERIALS ONLINE:

1. Course template
2. Tutorial question bank
3. Tech talk topics
4. Open end experiments
5. Definitions and terminology
6. Assignments
7. Model question paper - I
8. Model question paper - II
9. Lecture notes
10. E-learning readiness videos (ELRV)
11. Power point presentation



INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal - 500 043, Hyderabad, Telangana

COURSE CONTENT

DIGITAL CIRCUITS								
Course Code	Category	Hours / Week			Credits	Maximum Marks		
AECD13	Core	L	T	P	C	CIA	SEE	Total
		3	-	-	3	40	60	100
Contact Classes: 48	Tutorial Classes: Nil	Practical Classes: Nil			Total Classes: 48			
Prerequisite: Electronic Devices and circuits, Digital System Design								

I. COURSE OVERVIEW:

This course intended to logic gates, various logic families. Design of digital circuits using logic gates, combinational circuits and sequential circuits. Apply op-amp characteristics to design analog to digital converters and digital to analog converters. Classification and characteristics of memories such as Read-only memory, Random access memory and programmable logic devices such as programmable logic array and programmable array logic

II. COURSES OBJECTIVES:

The students will try to learn

- The fundamentals of number systems, Boolean algebra and representation of switching functions using Boolean expressions and their minimization techniques.
- The combinational and sequential logic circuits to design various complex switching devices, and their realizations.
- The programmable logic devices, Semiconductor memories and their use in realization of switching functions.
- The exploration of the logic families and semiconductor memories.

III. COURSE OUTCOMES:

At the end of the course students should be able to:

- CO1 Understand the different forms of number representations and binary codes in digital logic circuits
- CO2 Make use of Boolean postulates, theorems and k-map for obtaining minimized Boolean expressions
- CO3 Construct the synchronous and asynchronous modules using flip-flops used for memory storing applications
- CO4 Utilize the functionality and characteristics of flip-flops and latches for designing sequential circuits
- CO5 Describe the characteristics of the logic family and PLDs to enhance the design skills in digital integrated circuits.
- CO6 Extend the knowledge of memories and programmable logic devices for understanding the architectural blocks of FPGA.

IV. COURSE CONTENT:

MODULE - I: LOGICS IMPLIFICATION AND COMBINATIONAL LOGIC DESIGN(10)

Review of decimal, binary, octal and hexadecimal number system and conversions, Review of Boolean Algebra and De Morgan's Theorem, SOP & POS forms, Canonical forms, Karnaugh maps up to 6 variables, Binary codes, Code Conversion

MODULE –II: MSI DEVICES(10)

MSI devices like Comparators, Multiplexers, Encoder, Decoder, Driver & Multiplexed Display, Half and Full Adders, Subtractors, Full adder using Half adders, Full subtractor using Half subtractors, Serial and Parallel Adders, BCD Adder, Barrel shifter and ALU.

MODULE –III: SEQUENTIAL LOGIC DESIGN (10)

Building blocks like S-R, T FF, D FF, JK and Master-Slave JK FF, Edge triggered FF, Characteristics and Excitation of SR, JK, T, D FF, Ripple and Synchronous counters, Shift registers.

Finite state machines, Design of synchronous FSM, Algorithmic State Machines charts. Designing synchronous circuits like Pulse train generator, Pseudo Random Binary Sequence generator, Clock generation.

MODULE –IV: LOGIC FAMILIES AND SEMICONDUCTOR MEMORIES (09)

TTL NAND gate, Specifications, Noise margin, Propagation delay, fan-in, fan-out, Tristate TTL, ECL, CMOS families and their interfacing, Memory elements, Concept of Programmable logic devices like FPGA. Logic implementation using Programmable Devices.

MODULE-V: SEMICONDUCTOR MEMORIES AND PROGRAMMABLE LOGIC DEVICES (09)

Memory organization and operation, expanding memory size, classification and characteristics of memories, sequential memory, read only memory (ROM), read and write memory (RAM), content addressable memory (CAM), charge de coupled device memory (CCD), commonly used memory chips, ROM as a PLD, Programmable logic array, Programmable array logic, complex Programmable logic devices (CPLDS), Field Programmable Gate Array (FPGA).

V. TEXT BOOKS:

1. P Jain, "Modern Digital Electronics", McGraw Hill Education, 2009.
2. M M Mano, "Digital logic and Computer design", Pearson Education India, 2016.

VI. REFERENCE BOOKS:

1. A Kumar, "Fundamentals of Digital Circuits", Prentice Hall India, 2016.

VII. ELECTRONIC RESOURCES:

1. NPTEL :: NOC : Digital Electronic Circuit
<https://www.youtube.com/playlist?list=PLbRMhDVUMnge4gDT0vBWjCb3Lz0HnYKkX>

VIII. MATERIALS ONLINE:

1. Course template
2. Tutorial question bank
3. Tech talk topics
4. Open end experiments
5. Definitions and terminology
6. Assignments
7. Model question paper - I
8. Model question paper - II
9. Lecture notes
10. E-learning readiness videos (ELRV)
11. Power point presentation



INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal - 500 043, Hyderabad, Telangana

COURSE CONTENT

COMPLEX ANALYSIS AND PROBABILITY DISTRIBUTIONS

IV Semester: EEE

Course Code	Category	Hours/Week			Credits	Maximum Marks		
AHSD13	Foundation	L	T	P	C	CIA	SEE	Total
		3	-	-	3	40	60	100
Contact Classes:48	Tutorial Classes: Nil	Practical Classes: Nil				Total Classes: 48		
Prerequisite: Basic Principles of Algebra and Calculus								

Prerequisite: Basic Principles of Algebra and Calculus

I. COURSE OVERVIEW:

The course focuses on more advanced Engineering Mathematics topics which provide with the relevant mathematical tools required in the analysis of problems in engineering and scientific professions. The course includes complex functions and differentiation, complex integration, power series expansion of complex function and Probability of single random variables with its distributions. The mathematical skills derived from this course form a necessary base to analytical and design concepts encountered in the program.

II. COURSE OBJECTIVES:

The students will try to learn:

- I The applications of complex variable in two dimensional complex potential theories.
- II The fundamental calculus theorems and criteria for the independent path on contour integral used in problems of engineering
- III The statistical tools which are essential for translating an engineering problem into probability model.

III. COURSE OUTCOMES:

At the end of the course students should be able to:

- CO1 Identify the fundamental concepts of analyticity and differentiability for finding complex conjugates of complex transformations
- CO2 Apply integral theorems of complex analysis and its consequences for the analytic function with derivatives of all orders in simple connected region
- CO3 Extend the Taylor and Laurent series for expressing the function in terms of complex power series
- CO4 Apply Residue theorem for computing definite integrals by using the singularities and poles of real and complex analytic functions over closed curves.
- CO5 Apply the role of random variables and types of random variables, expected values of the discrete and continuous random variables under randomized probabilistic conditions
- CO6 Apply the parameters of random variable Probability distributions such as Binomial, Poisson by using their probability functions,

IV. COURSE SYLLABUS:

MODULE-I: FUNCTIONS OF A COMPLEX VARIABLE (10)

Functions of a complex variable; concept of limits, continuity and differentiability of complex function, analyticity, Cauchy-Riemann equations (without proof); harmonic functions, constructions of analytic function, Milne-Thomson method.

MODULE-II: COMPLEX INTEGRATION (10)

Line integral: Evaluation along a path and by indefinite integration; Cauchy's integral theorem; Cauchy's integral formula; generalised Cauchy integral formula,

MODULE-III: POWER SERIES OF COMPLEX FUNCTIONS (09)

Expansion of the complex function in Taylor's series, Maclaurin's series and Laurents's series (all theorems without proof); Singularities.

Residues: Cauchy Residue Theorem (without proof); Evaluation of integrals of the type $\int_0^{2\pi} f(\cos\theta, \sin\theta)d\theta$ by residues.

MODULE-IV: RANDOM VARIABLES (09)

Random variables: Discrete and continuous random variables, Probability distribution; Probability mass function and probability density functions,

MODULE-V: PROBABILITY DISTRIBUTION (10)

Binomial distribution; Mean and variances of Binomial distribution, Poisson distribution: Poisson distribution as a limiting case of Binomial distribution, mean and variance of Poisson distribution. Normal distribution; Mean, Variance, Mode, Median of Normal distribution

V. TEXT BOOKS:

1. Erwin Kreyszig, "Advanced Engineering Mathematics", John Wiley and Sons Publishers, 9th edition, 2014.
2. B. S. Grewal, Higher Engineering Mathematics, 44/e, Khanna Publishers, 2017.

VI. REFERENCE BOOKS:

1. R. K. Jain and S. R. K. Iyengar, Advanced Engineering Mathematics, 3rd ed, Narosa Publications, 5th edition, 2016.
2. S. C. Gupta, V. K. Kapoor, "Fundamentals of Mathematical Statistics", S. Chand and Co., 10th edition, 2000.
3. N.P. Bali and Manish Goyal, A text book of Engineering Mathematics, Laxmi Publications, Reprint, 2008.
4. Richard Arnold Johnson, Irwin Miller and John E. Freund, "Probability and Statistics for Engineers", Prentice Hall, 8th edition, 2013.

VII. ELECTRONIC RESOURCES:

1. http://www.efunda.com/math/math_home/math.cfm
2. <http://www.ocw.mit.edu/resources/#Mathematics>
3. <http://www.sosmath.com>
4. <http://www.mathworld.wolfram.com>

VIII. MATERIALS ONLINE:

1. Course template
2. Tutorial question bank
3. Tech talk topics
4. Open end experiments
5. Definitions and terminology
6. Assignments
7. Model question paper - I
8. Model question paper - II
9. Lecture notes
10. E-learning readiness videos (ELRV)
11. Power point presentation



INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal - 500 043, Hyderabad, Telangana

COURSE CONTENT

AC MACHINES LABORATORY								
IV Semester: EEE								
Course Code	Category	Hours/Week			Credits	Maximum Marks		
AEED13	Core	L	T	P	C	CIA	SEE	Total
		0	0	2	1	40	60	100
Contact Classes: Nil	Tutorial Classes: Nil	Practical Classes: 45			Total Classes: 45			

Pre-requisite: Electrical Circuits, Linear Algebra and Calculus

I. COURSE OVERVIEW:

This course is intended to train the students on alternating current machines. It provides hands-on experience by conducting various direct and indirect tests on transformers, synchronous and asynchronous machines to analyze the characteristics of ac machines and separate various losses. This course also enables to develop skills to select, install, operate, and maintain various types of ac machines and transformers.

II. COURSE OBJECTIVES:

The students will try to learn:

- I The elementary experimental and modeling skills for handling problems with electrical machines in industries and domestic applications
- II The operation of AC machines and its role in power transmission and generating stations
- III The automation concepts through programmable logic controllers to control the speed and starting current.

III. COURSE OUTCOMES:

After successful completion of the course, students should be able to:

- CO1 Select suitable testing strategies for evaluating the performance characteristics of transformers.
- CO2 Determine the performance parameters of induction motor by conducting direct and indirect tests.
- CO3 Explain the parallel operation of alternators for load sharing under various loading conditions.
- CO4 Distinguish the synchronous impedance and ampere turns methods for the computation of voltage regulation of an alternator.
- CO5 Estimate the voltage and current swings in salient pole alternator for determination of direct and quadrature axis reactance.
- CO6 Apply programmable logic controllers for limiting the starting current of poly phase induction motors.

IV. COURSE CONTENT:

DO's

1. Once the operation is completed pull the plug itself rather chord attached to it.
2. To repair the equipment switch-off the supply and go on.
3. To operate the equipment on supply, see that hands are dry, if that is not possibly hide the hand in the pockets.
4. If a person comes in contact with current unexpectedly don't touch the person with hands but immediately use any insulator material and shut down the power
5. If water is nozzles on the equipment, immediately shunt down the power using circuit breaker or pull out the plug.
6. Use the connecting wires of good continuity, short circuit of connecting wire leads damage of circuit parameters

DON'Ts

1. Do not wear loose clothing and do not hold any conducting materials in contact with skin when the power is on.
2. Do not pull out the connections until unless all the currents are dead.
3. Do not wait for switches to de-magnetize when there is a delay but pull out the plug.
4. Do not overload the circuit by plugging in too many appliances.
5. If you are mentally and physically stressed don't operate the power equipment.
6. Never operate the equipment under wet conditions.
7. Do not interconnect two or more wires, take appropriate length of wire.

SAFETY NORMS

7. The lab must be equipped with fire extinguisher.
8. See that the connections are made tight.
9. Use single plug for each equipment.
10. Cover the body completely to avoid arc effect.
11. To change the connections during the experiment, switch off the supply and carry on.
12. Used equipment may get heated, so take care handling the equipment after it is used.
13. Do the wiring, all set ups and check the circuit connections before the supply is on

IV. COURSE CONTENT:

EXERCISES FOR ELECTRICAL AC MACHINES LABORATORY

Note: Students are encouraged to bring their own laptops for laboratory practice session

1. OPEN AND SHORT CIRCUIT TEST ON A SINGLE - PHASE TRANSFORMER

Conduct the experiment to determine the circuit parameters at rated voltage and rated current.

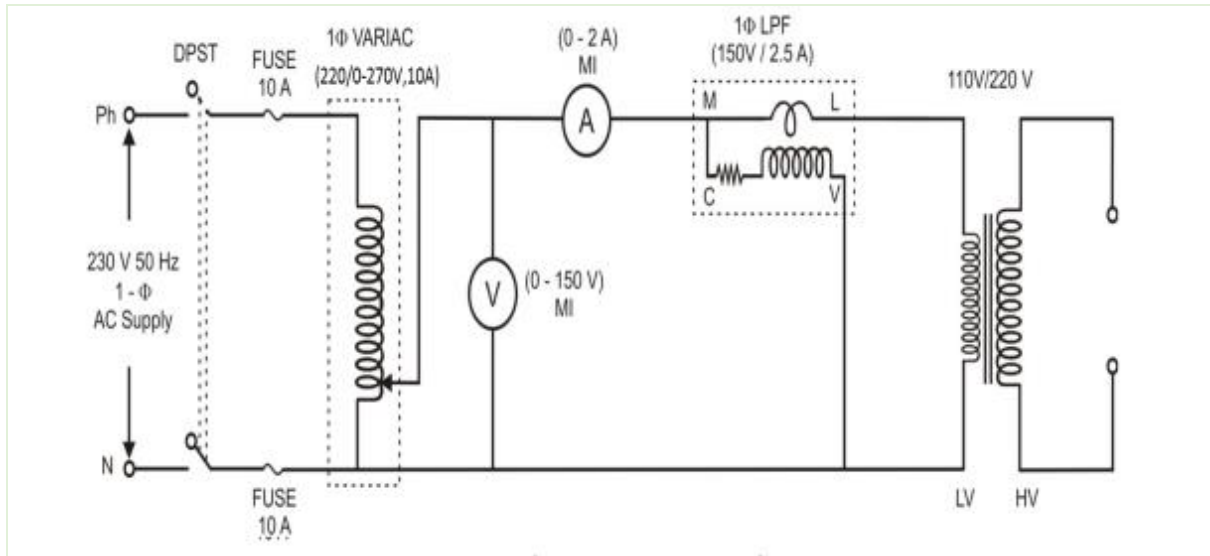


Figure 1 - Open circuit test Circuit

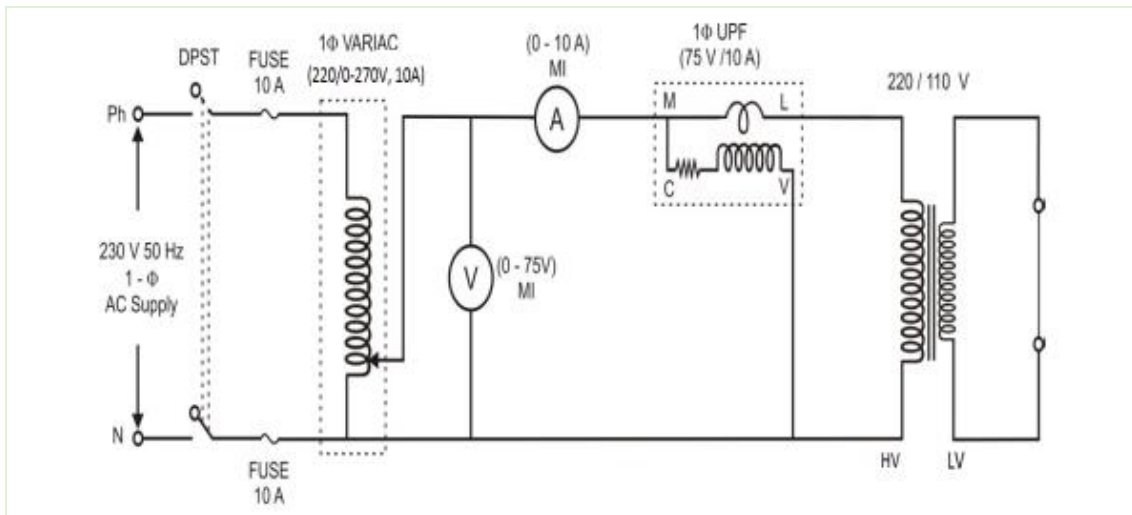


Figure 1.1 - Short circuit test Circuit

Try

1. From the Open circuit characteristics calculate the open circuit voltage and determine the performance using OCC from Figure 1.
2. Using short circuit characteristics calculate the short circuits current and determine the performance using SCC from Figure 2.

2. SUMPNERS TEST ON A PAIR OF SINGLE - PHASE TRANSFORMERS

Conduct an experiment on pair of single - phase transformers to determine the regulation and efficiency.

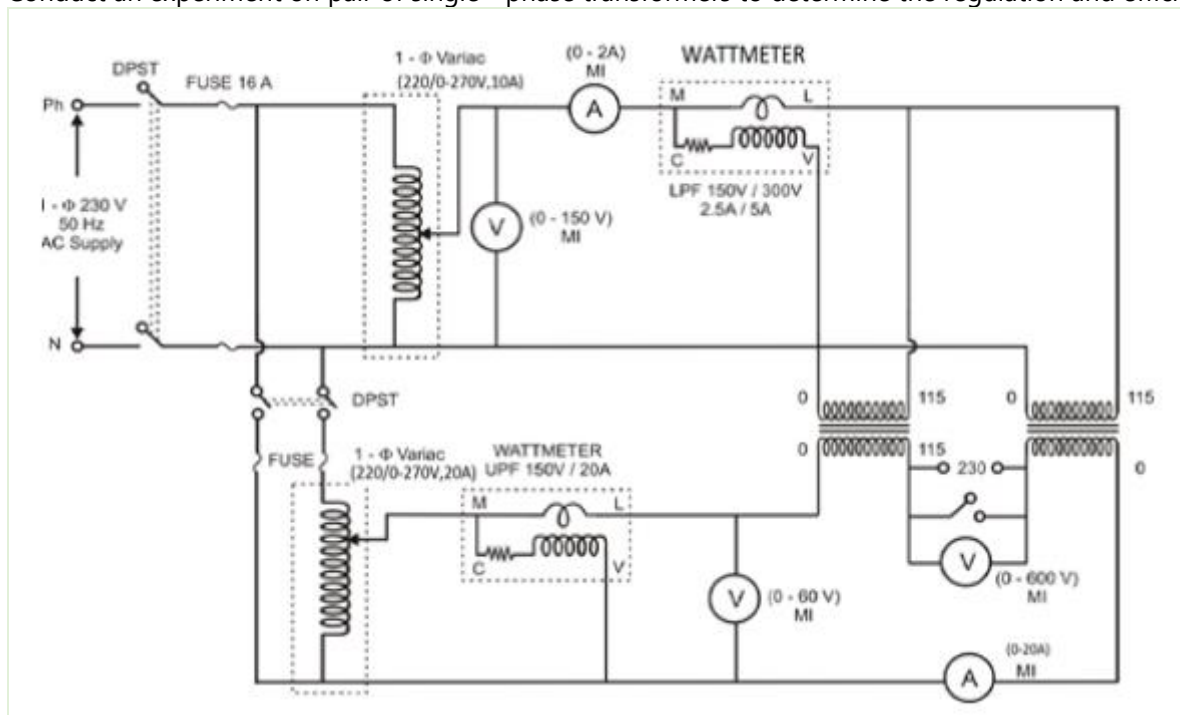


Figure 2 – Sumpner's test on a pair of single phase Transformers

Try

4. Calculate the circuit parameters like currents, voltages and power from Figure 2
5. Determine the regulation and efficiency by using the circuit parameters obtained.

3. LOAD TEST ON SINGLE PHASE TRANSFORMERS

Conduct an experiment to determine efficiency by performing load test on a single phase transformer.

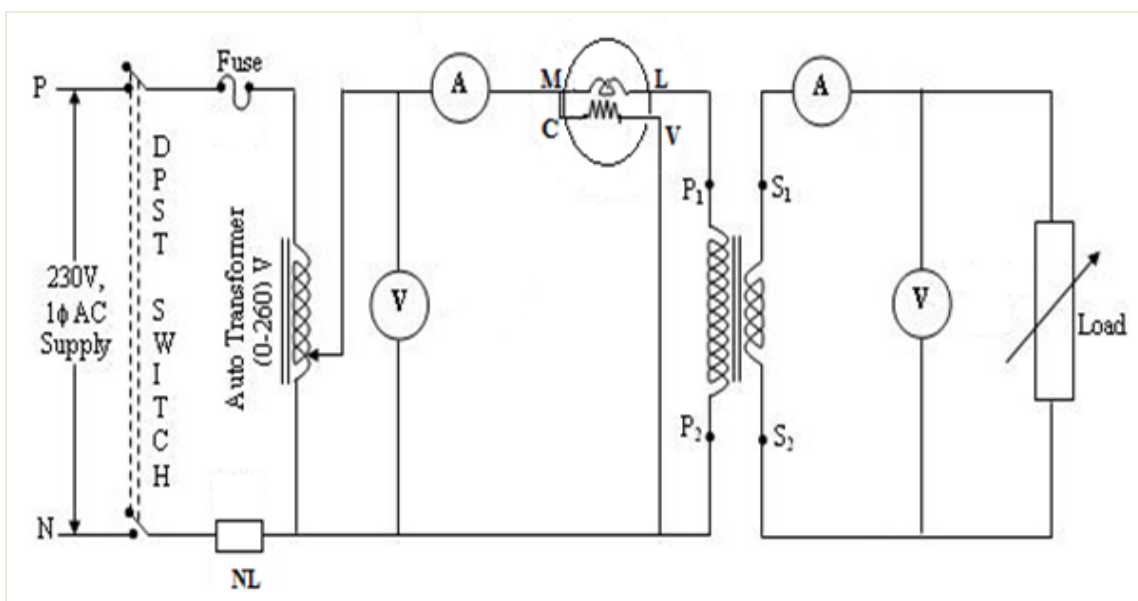
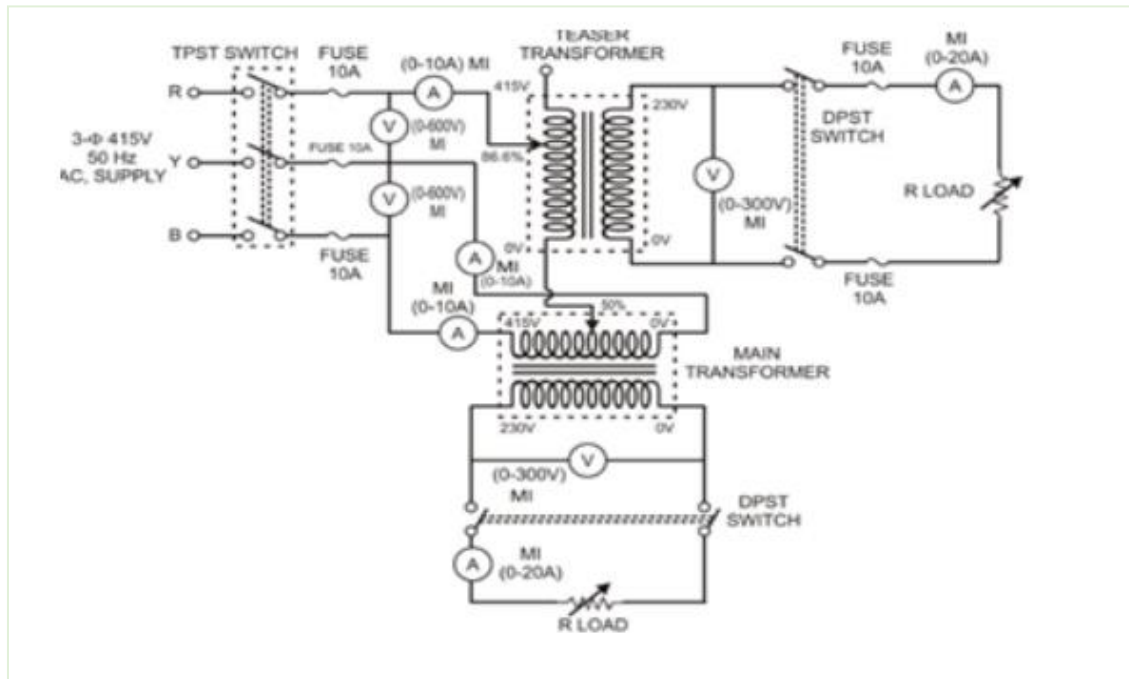


Figure 3 - LOAD TEST ON SINGLE PHASE TRANSFORMERS

Try

1. Calculate the different currents and voltages by using voltmeters and ammeters from Figure 3.
2. Calculate the efficiency by performing the test on transformer

4. SCOTT CONNECTION OF TRANSFORMERS



Conduct an experiment to Convert three phase to two phase using single phase transformers

Figure 4 - SCOTT CONNECTION OF TRANSFORMERS

Try

1. Calculate the different currents and voltages by using voltmeters and ammeters from Figure 4.
2. By increasing the load current on transformers, Calculate load currents and voltages.

5. SEPARATION OF CORE LOSSES IN SINGLE PHASE TRANSFORMER

Conduct an experiment to determine the eddy current losses and hysteresis losses in single phase transformer

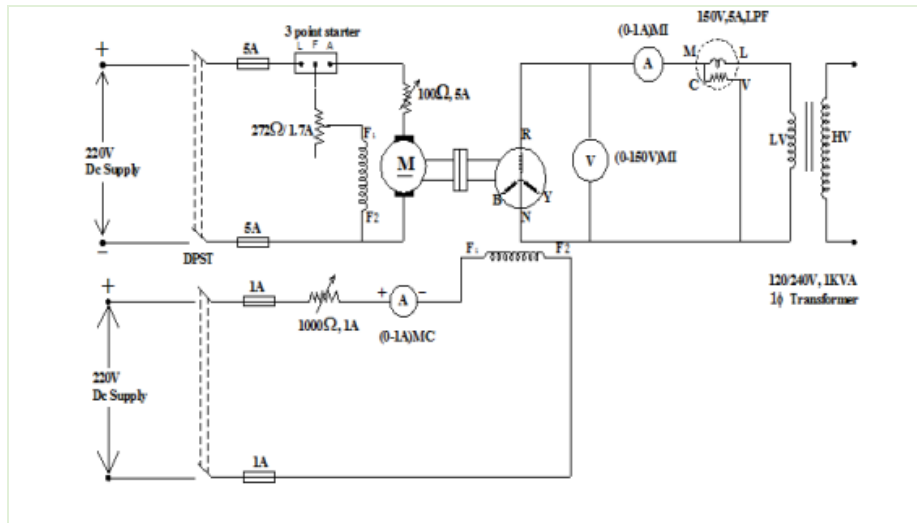


Figure 5 - SEPARATION OF CORE LOSSES IN SINGLE PHASE TRANSFORMER

Try

1. Calculate the value of eddy current losses with respect to current and voltage and power parameters' obtained at different frequencies by maintain V/F as constant from Figure 5.
2. Calculate the value of hysteresis losses with respect to current and voltage, power parameters' obtained at different frequencies by maintain V/F as constant.

6. HEAT RUN TEST ON SINGLE PHASE TRANSFORMERS

Conduct an experiment to check whether the oil and winding temperatures of the transformer meet the values specified in the standard and technical projects.

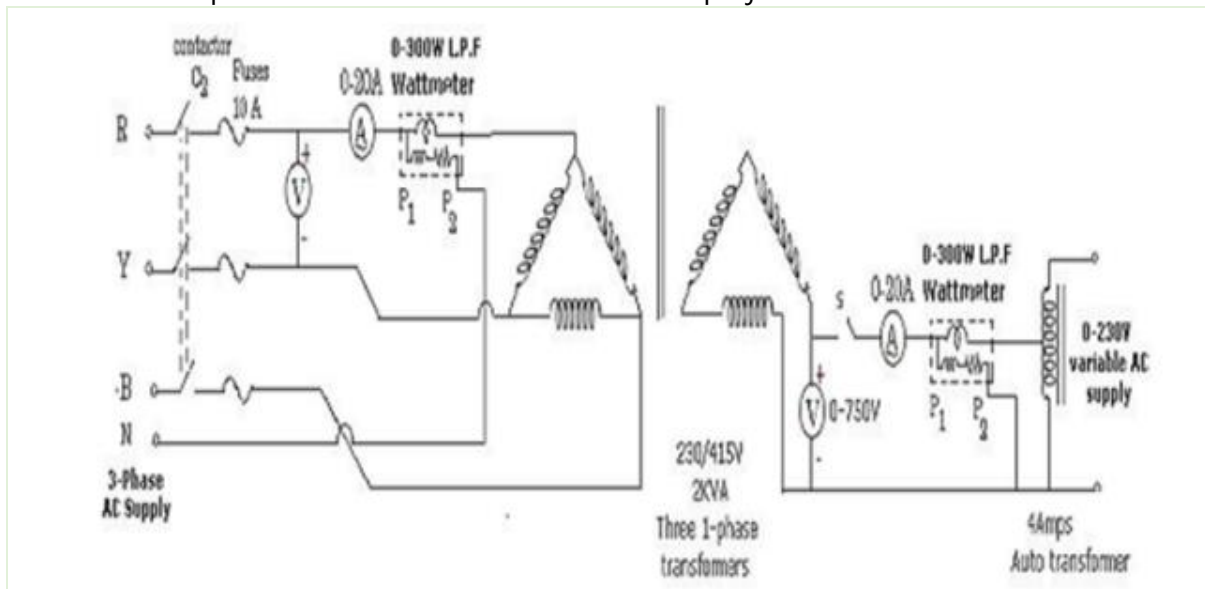


Figure 6 - HEAT RUN TEST ON SINGLE PHASE TRANSFORMERS

Try

1. Calculate the temperature rise by determining voltage, current and power from Figure 6.
2. Conduct the polarity test and connect them as per the dot Convention when voltmeter indicates high value.

7. BRAKE TEST ON THREE PHASE SQUIRREL CAGE INDUCTION MOTOR

Calculate the rotor and stator losses (mainly copper losses) by conducting blocked rotor test.

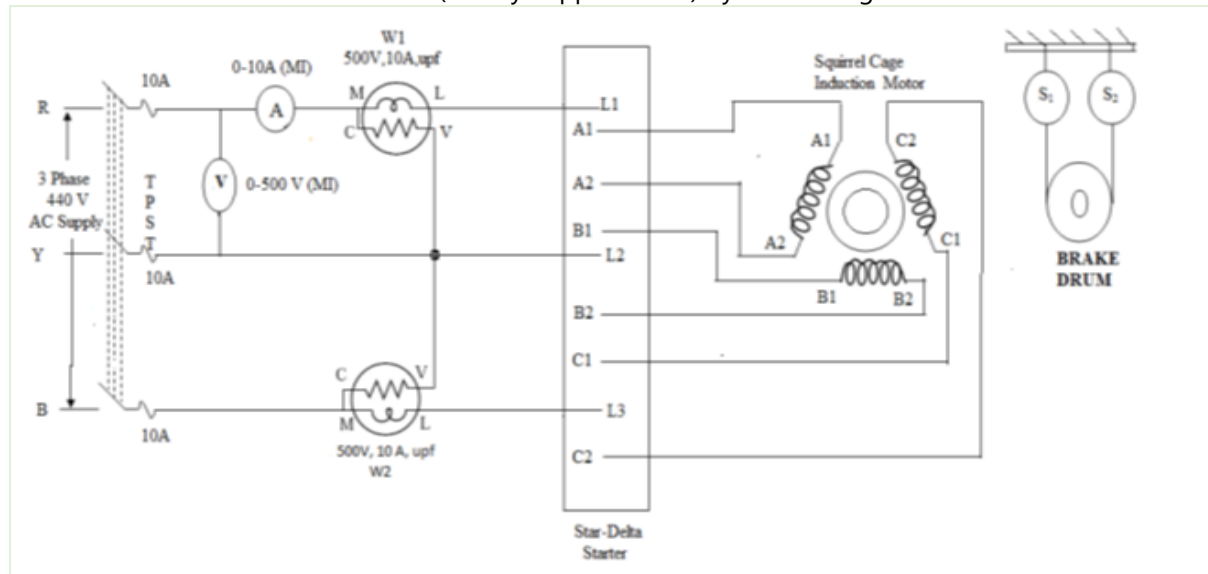


Figure 7 - BRAKE TEST ON THREE PHASE SQUIRREL CAGE INDUCTION MOTOR

Try

1. Calculate the voltage, current and power values at no-load condition from Figure 7.
2. By applying mechanical load, calculate all meter readings including speed of induction motor.
3. Plot the performance characteristics of three phase induction motor

8. CIRCLE DIAGRAM OF THREE PHASE SQUIRREL CAGE INDUCTION MOTOR

Plot the circle diagram and predetermine the efficiency and losses of three phase squirrel cage induction motor.

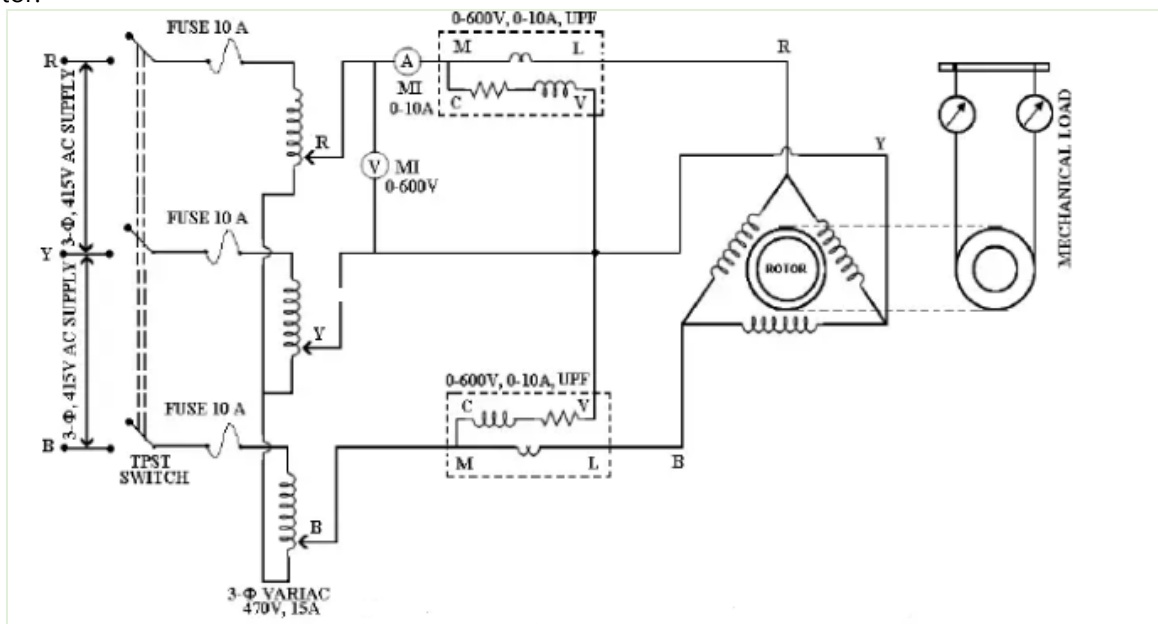


Figure 8 - CIRCLE DIAGRAM OF THREE PHASE SQUIRREL CAGE INDUCTION MOTOR

Try

1. Calculate the efficiency of Squirrel cage IM under different load conditions from Figure 8.
2. Plot the circle diagram by determining the currents and voltages connected in circuit.

9. REGULATION OF ALTERNATOR BY EMF METHOD

Determine the regulation of alternator using synchronous impedance method

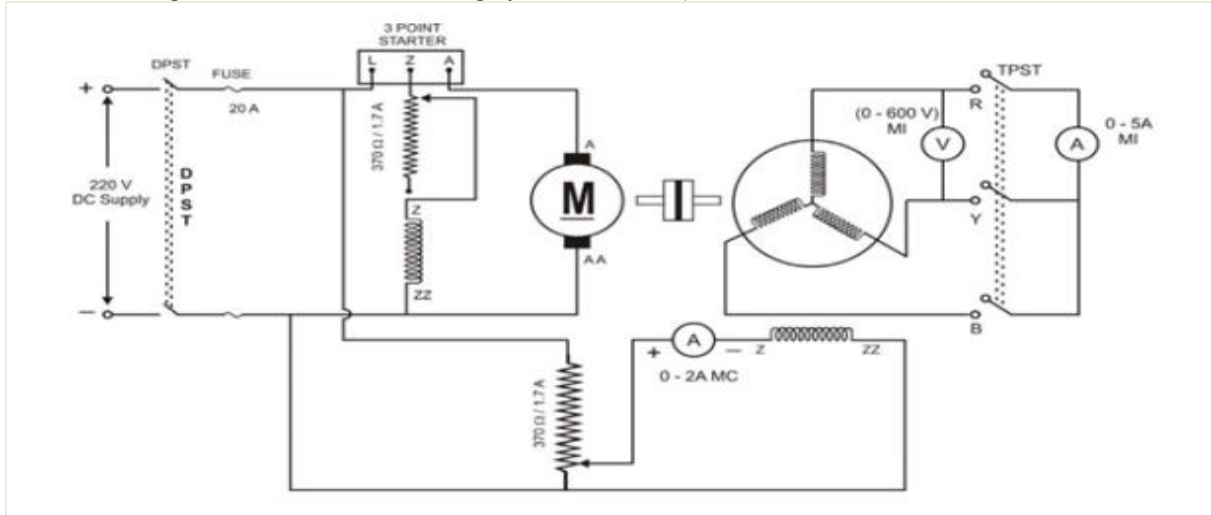


Figure 9 - REGULATION OF ALTERNATOR BY EMF METHOD

Try

1. Calculate the OCV from the circuit from Figure 9.
2. Draw the OCC curve
3. Determine the voltage regulation by using EMF method.

10. REGULATION OF ALTERNATOR BY MMF METHOD

Determine the regulation of alternator using amperes turns method

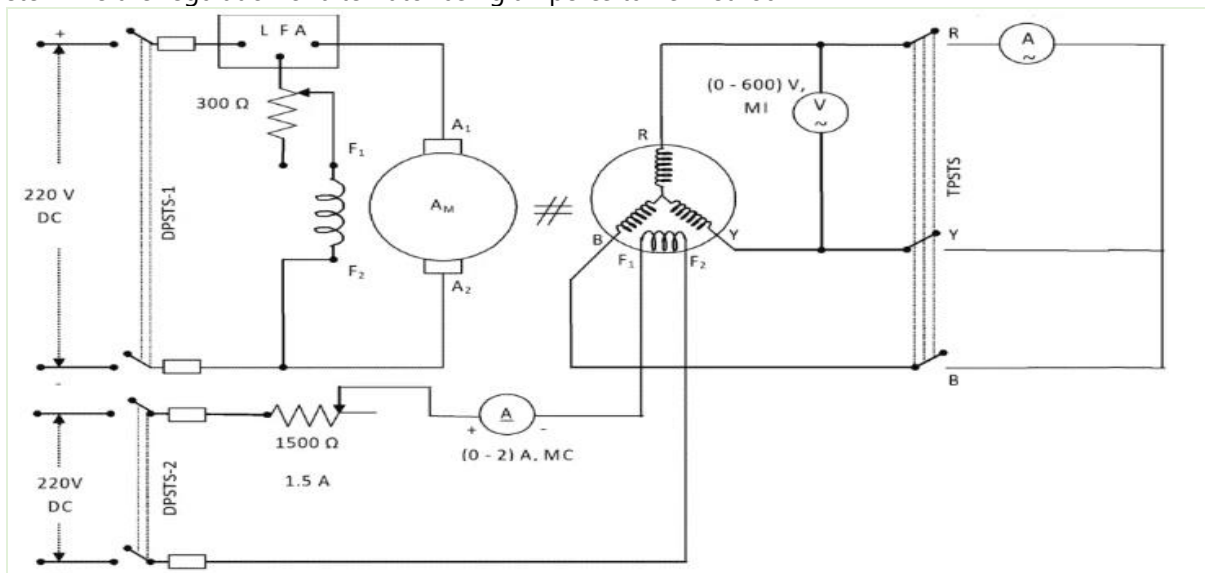


Figure 10 - REGULATION OF ALTERNATOR BY MMF METHOD

Try

1. Calculate the Short circuit current from the circuit in Figure 10.
2. Draw the SCC curve
3. Determine the voltage regulation by using MMF method.

11. SLIP TEST ON THREE PHASE SALIENT POLE SYNCHRONOUS MOTOR

Determination of X_d and X_q in a three phase salient pole synchronous motor.

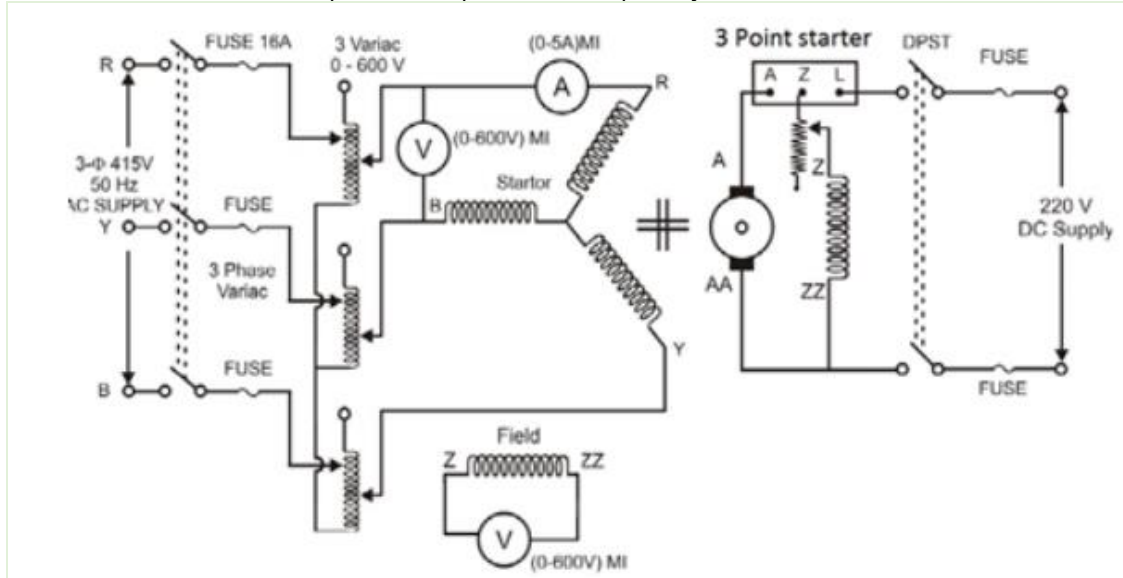


Figure 11 - SLIP TEST ON THREE PHASE SALIENT POLE SYNCHRONOUS MOTOR

Try

1. Calculate the direct axis reactance (X_d) by calculating min voltage and max current from Figure 11
2. Calculate the quadrature axis reactance (X_q) by calculating max voltage and min current.

12. V AND INVERTED V CURVES OF SYNCHRONOUS MOTOR

Plot V and inverted V curves to study the effect of power factor in synchronous motor.

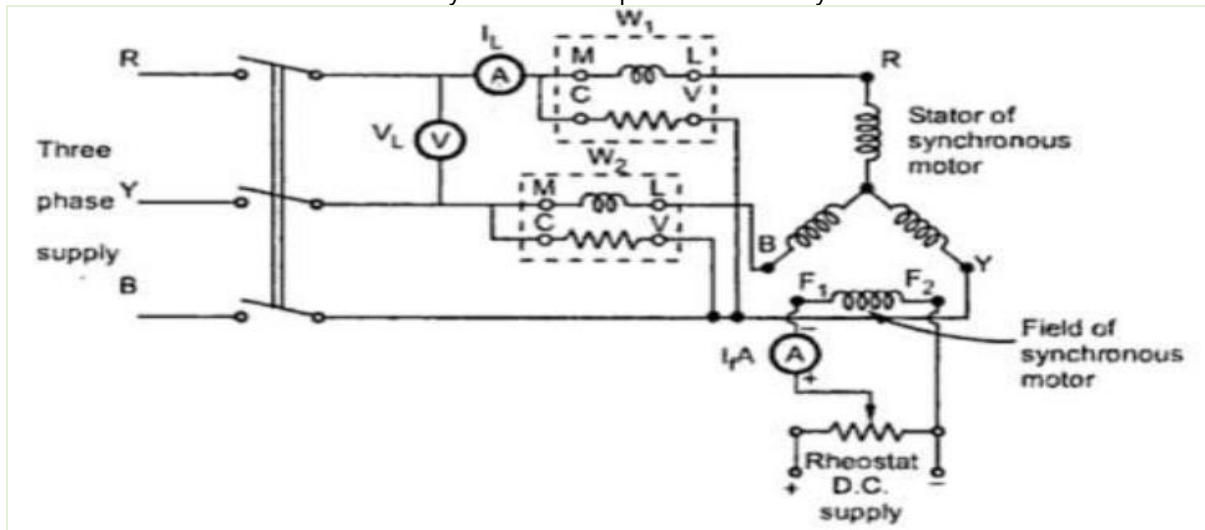


Figure – 12 V AND INVERTED V CURVES OF SYNCHRONOUS MOTOR

Try

1. For different Power factors plot the V curve and inverted V curves for lagging, leading and Unity PF from Figure 12.

13. EQUIVALENT CIRCUIT PARAMETERS OF SINGLE PHASE INDUCTION MOTOR

Determine the equivalent circuit parameters of a single phase induction motor

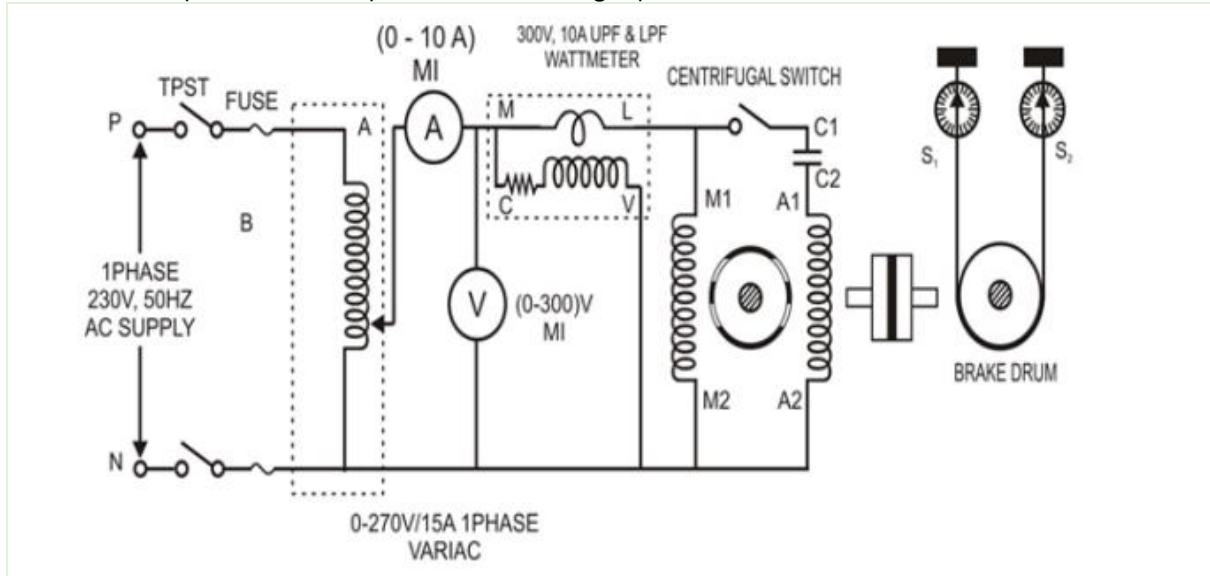


Figure – 13 EQUIVALENT CIRCUIT PARAMETERS OF SINGLE PHASE INDUCTION MOTOR

Try

1. Calculate the equivalent circuit parameters like current, voltage and power under various loads from Figure 13
2. Draw the equivalent circuit diagram of induction motor.

14. STARTING AND SPEED CONTROL OF INDUCTION MOTOR USING PLC

Implementation of direct online (DOL) starter for three phase induction motor using Programmable Logic Controllers.

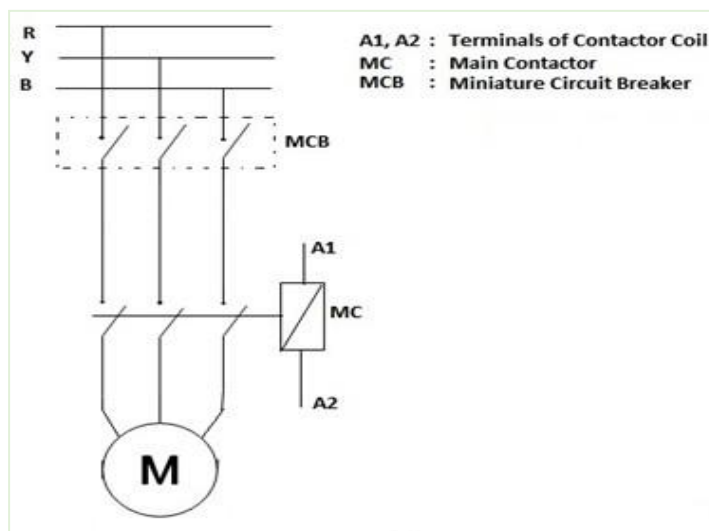


Figure – 14 STARTING OF INDUCTION MOTOR USING PLC

Try

1. Analyze about the components and operation of direct online starter using ladder diagram in terms of concepts.
2. Conclude the above obtained results with wiring the Programmable Logic Controller and Human Machine Interface.

Automatic Forward and Reverse Control of Motor

Implement automatic forward and reverse control of three phase squirrel cage induction motor for milling operation using PLC.

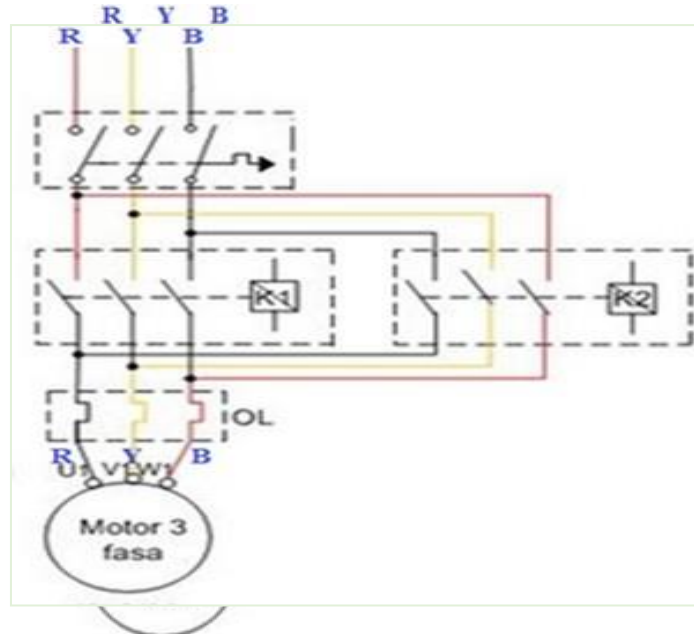


Figure – 14.1 SPEED CONTROL OF INDUCTION MOTOR USING PLC

Try

1. Analyze the forward and reverse control operation of three phase induction motor for Circuit using ladder diagram in terms of concepts.
2. Conclude the above obtained results with wiring the Programmable Logic Controller and Human Machine Interface.

V. REFERENCE BOOKS:

1. P S Bimbhra, “Electrical Machines”, Khanna Publishers, 2nd edition, 2008.
2. M G Say, E O Taylor, “Direct Current Machines”, Longman Higher Education, 1st edition, 1985.
3. Hughes, “Electrical Technology”, Prentice Hall, 10th edition, 2015.
4. Nesimi Ertugrul, “LabVIEW for Electric Circuits, Machines, Drives, and Laboratories”, Prentice Hall, 1st edition, 2002.
5. Gupta, Gupta & John, “Virtual Instrumentation Using LabVIEW”, Tata McGraw-Hill, 1st edition, 2005.

VI. WEB REFERENCES:

1. <https://www.ee.iitkgp.ac.in>
2. <https://www.citchennai.edu.in>
3. <https://www.iare.ac.in/>

VII. MATERIALS ONLINE

1. Course template
2. Lab manual



INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal - 500 043, Hyderabad, Telangana

COURSE CONTENT

CONTROL SYSTEMS LABORATORY								
III Semester: EEE								
Course Code	Category	Hours / Week			Credits	Maximum Marks		
AEED14	Core	L	T	P	C	CIA	SEE	Total
		-	-	2	1	40	60	100
Contact Classes: Nil	Tutorial Classes: Nil	Practical Classes: 45			Total Classes: 45			

Prerequisite: There are no prerequisites to take this course

I. COURSE OVERVIEW:

The Control Systems laboratory course is indeed to train the students practically on the modelling, analysis and design of linear feedback control systems. This course deals with modelling of dynamical systems, and the control components and designing the compensator. The hands on training in the laboratory enable students to apply and modelling control principles in various areas of industrial applications.

II. COURSES OBJECTIVES:

The students will try to learn

- I The estimation of stability of dynamical systems using Digital simulation.
- II The various techniques of modeling and analyzing system's performance.
- III Design the time and frequency response of system by both classical and modern techniques

III. COURSE OUTCOMES:

At the end of the course students should be able to:

- CO1 Make use of the knowledge of digital simulation tool for system analysis with different standard Inputs.
- CO2 Model the dynamic systems in transfer function using digital simulation tool and validate the performance characteristics of motors.
- CO3 Analyze and select various electronics devices for improving system performance along with tuning mechanism in virtual environment.
- CO4 Experiment the types of compensation techniques for improving the system's accuracy.
- CO5 Analyze the system's stability in time and frequency domain by computing gain and phase Margin.

IV. COURSE CONTENT:

DO's

1. Once the operation is completed pull the plug itself rather chord attached to it.
2. To repair the equipment switch-off the supply and go on.
3. To operate the equipment on supply, see that hands are dry, if that is not possibly hide the hand in the pockets.
4. If a person comes in contact with current unexpectedly don't touch the person with hands but immediately use any insulator material and shut down the power (like leather belts, wood and plastic bars etc.).
5. If water is nozzles on the equipment, immediately shunt down the power using circuit breaker or pull out the plug.
6. Use the connecting wires of good continuity, short circuit of connecting wire leads damage of circuit parameters

DON'Ts

1. Do not wear loose clothing and do not hold any conducting materials in contact with skin when the power is on.
2. Do not pull out the connections until unless all the currents are dead.
3. Do not wait for switches to de-magnetize when there is a delay but pull out the plug.
4. Do not overload the circuit by plugging in too many appliances.
5. If you are mentally and physically stressed don't operate the power equipment.
6. Never operate the equipment under wet conditions.
7. Do not interconnect two or more wires, take appropriate length of wire.

SAFETY NORMS

14. The lab must be equipped with fire extinguisher.
15. See that the connections are made tight.
16. Use single plug for each equipment.
17. Cover the body completely to avoid arc effect.
18. To change the connections during the experiment, switch off the supply and carry on.
19. Used equipment may get heated, so take care handling the equipment after it is used.
20. Do the wiring, all set ups and check the circuit connections before the supply is on

IV. COURSE CONTENT:

EXERCISES FOR CONTROL SYSTEMS LABORATORY

Note: Students are encouraged to bring their own laptops for laboratory practice session

Getting Started Exercises

Introduction to MATLAB

Identify the symbols, tool kits and connections in Simulink environment for computing the quantities associated with electrical circuits.

MATLAB is a powerful software environment widely used in various fields, including electrical engineering and circuit analysis. It provides a range of symbols, toolkits, and connections related to electrical circuits to assist engineers and researchers in designing, analyzing, and simulating electrical circuits. Here's an introduction to some of the key features and tools MATLAB offers in this regard:

1. **Symbolic Math Toolbox:** MATLAB's Symbolic Math Toolbox allows you to perform symbolic computations, which are crucial for electrical circuit analysis. You can define symbolic variables for circuit components like resistors, capacitors, and inductors and manipulate algebraic equations symbolically. This is useful for solving circuit equations and obtaining transfer functions.
2. **Simulink:** Simulink is a graphical environment within MATLAB for modeling, simulating, and analyzing dynamic systems, including electrical circuits. It provides a vast library of pre-built electrical components and blocks, making it easier to create circuit models and simulate their behavior.
3. **Control System Toolbox:** This toolbox contains tools and functions for analyzing and designing control systems, which are essential for understanding the behavior of electrical circuits. You can use it to design controllers, analyze stability, and simulate the response of electrical systems.
4. **Electrical Circuit Analysis Toolbox:** There are also third-party toolboxes available for MATLAB, such as the Electrical Circuit Analysis Toolbox, that provide specialized tools for electrical circuit analysis. These toolboxes often include features like circuit simulation, AC/DC analysis, and more.
5. **SimPowerSystems (formerly known as SimElectronics):** SimPowerSystems is a specialized toolbox in Simulink for modeling and simulating electrical circuits and systems. It includes a wide range of electrical components, such as transformers, generators, and motors, allowing you to create detailed electrical system models.
6. **Simscape Electrical:** Simscape Electrical is another Simulink toolbox that focuses on modeling and simulating electrical power systems. It includes components like power electronics devices, electrical machines, and power generation and distribution elements.
7. **Signal Processing Toolbox:** This toolbox is valuable for analyzing signals within electrical circuits. You can use it for tasks like filtering noisy signals, performing Fourier analysis, and extracting important information from sensor data.
8. **Data Acquisition Toolbox:** If you're working with hardware experiments or data acquisition systems, this toolbox can help you connect MATLAB to external hardware, making it easier to acquire and analyze data from real-world electrical circuits.
9. **Instrument Control Toolbox:** This toolbox enables you to communicate with and control external instruments like oscilloscopes, signal generators, and multimeters, which is beneficial for experimental work in electrical engineering.

10. **Custom Functions and Scripts:** MATLAB allows you to create custom functions and scripts to model and analyze specific aspects of electrical circuits. You can define your own functions for circuit equations, transfer functions, and more.

Creating syntax programs for electrical circuit analysis in MATLAB often involves using built-in functions, toolboxes, and custom code to define circuits, simulate their behavior, and analyze results. Below are some examples of syntax programs for common tasks in electrical circuit analysis using MATLAB:

Hint

1. Calculate the Circuit Components and Equations:

```
% Define symbolic variables
syms V1 V2 R1 R2 C1 L1 s

% Define resistor values
R1 = 100; % Ohms
R2 = 200; % Ohms

% Define capacitor and inductor values
C1 = 0.1; % Farads
L1 = 0.5; % Henrys

% Define Kirchhoff's laws for an RC circuit
Eq1 = V1 - R1 * I1 - V2 == 0;
Eq2 = V2 - R2 * I2 - I1/C1 == 0;

% Define Laplace domain equations
Eq1_Laplace = laplace (Eq1, t, s);
Eq2_Laplace = laplace (Eq2, t, s);
```

2. Solve Circuit Equations:

```
% Solve the Laplace domain equations symbolically
I1_Laplace = solve (Eq1_Laplace, I1);
I2_Laplace = solve (Eq2_Laplace, I2);

% Inverse Laplace transform to get time-domain solutions
I1_time = ilaplace (I1_Laplace, s, t);
I2_time = ilaplace (I2_Laplace, s, t);
```

3. Simulate the Circuits in Simulink:

```
% Create a Simulink model for an RLC circuit
model = 'RLC_Circuit';
open_system(new_system(model));

% Add components and connections using Simulink blocks
add_block ('Simulink/Continuous/Resistor', [model '/R1']);
add_block ('Simulink/Continuous/Inductor', [model '/L1']);
add_block ('Simulink/Continuous/Capacitor', [model '/C1']);
add_block ('Simulink/Sources/Step', [model '/Step Voltage']);
add_line (model, 'Step Voltage/1', 'R1/1');
% Add connections as needed

% Set component values and simulation parameters
set_param ([model '/R1'], 'Resistance', '100');
```

```
set_param ([model '/L1'], 'Inductance', '0.5');
set_param ([model '/C1'], 'Capacitance', '0.1');
set_param (model, 'Stop Time', '5');

% Run the simulation
sim(model);
```

4. Analyze Frequency Response:

```
% Define a transfer function for an RC circuit
num = [1];
den = [R1*C1 1];
sys = tf (num, den);

% Bode plot for frequency response
bode(sys);
```

5. Evaluate the Data Analysis and Plotting:

```
% Import experimental data from a file
data = import data('experimental_data.csv');
time = data (: 1);
voltage = data (: 2);

% Plot voltage-time graph
plot (time, voltage);
xlabel ('Time (s)');
ylabel ('Voltage (V)');
title ('Voltage vs. Time');
```

Try:

a. Circuit Equations:

- i. Write down the Kirchhoff's Voltage Law (KVL) equation for a series RC circuit involving a resistor (R) and a capacitor (C). Assume a constant voltage source V_S .
- ii. Write down the Kirchhoff's Voltage Law (KVL) equation for a series RL circuit involving a resistor (R) and an inductor (L). Assume a time-varying current source $I_S(t)$.

b. MATLAB Code: Write MATLAB code to:

- i. Solve the KVL equation for the RC circuit (from question 2a) symbolically for V_C as a function of time, assuming initial conditions.
- ii. Create a Simulink model for the RL circuit (from question 2b) with appropriate components and connections. Set the simulation parameters and run a transient analysis for a specified time duration.

Hint: Save your MATLAB code as a separate script file (.m) for questions

c. Transfer Function:

- i. Define the transfer function $H(s)$ for the RC circuit in terms of Laplace variables.
- ii. Plot the Bode plot of $H(s)$ to analyze the frequency response of the RC circuit.

d. Data Analysis:

- i. Import experimental voltage and time data from a CSV file into MATLAB.
- ii. Plot the voltage-time graph and label the axes appropriately.

e. Circuit Toolbox:

Mention at least two MATLAB toolboxes that are useful for electrical circuit analysis. Explain briefly how

each toolbox can assist in circuit analysis.

- f.** If $R = 10$ Ohms and the current is increased from 0 to 10 A with increments of 2A, write a MATLAB program to generate a table of current, voltage and power dissipation.

1. TIME RESPONSE OF SECOND ORDER SYSTEM

To determine response of first order and second order systems for step input for several of constant 'K' using linear simulator unit and compare theoretical and practical results.

1.1. TIME RESPONSE OF SECOND ORDER SYSTEM IN MATLAB

Examine the time response of a given second order system with time domain specifications using MATLAB

To find the time domain specifications of a second-order system in MATLAB, you can use the "**stepinfo**" function, which provides information such as rise time, settling time, overshoot, etc. Here's an example with the necessary syntax:

Hint

```
% Define system parameters

Omega_n = 2; % Natural frequency
zeta = 0.7; % Damping ratio

% Create the transfer function
numerator = omega_n^2;
denominator = [1, 2*zeta*omega_n, omega_n^2];
sys = tf(numerator, denominator);

% Obtain time domain specifications
info = stepinfo(sys);

% Display time domain specifications
disp('Time Domain Specifications:');
disp(['Rise Time: ' num2str(info.RiseTime) ' seconds']);
disp(['Settling Time: ' num2str(info.SettlingTime) ' seconds']);
disp(['Overshoot: ' num2str(info.Overshoot) ' %']);
disp(['Peak Time: ' num2str(info.PeakTime) ' seconds']);
```

Try

1. MATLAB program for finding step, ramp parabolic input response for a closed-loop transfer function $G(s) = \frac{3}{s(s+4)}$
 2. MATLAB program to find the unit step response of a unity feedback control system with $G(s) = \frac{10}{s(s+3)}$ and $H(s) = 0.1s + 1$
-

1.2. TIME RESPONSE OF SECOND ORDER SYSTEM

Examine the time response of a given second order system with time domain specifications.

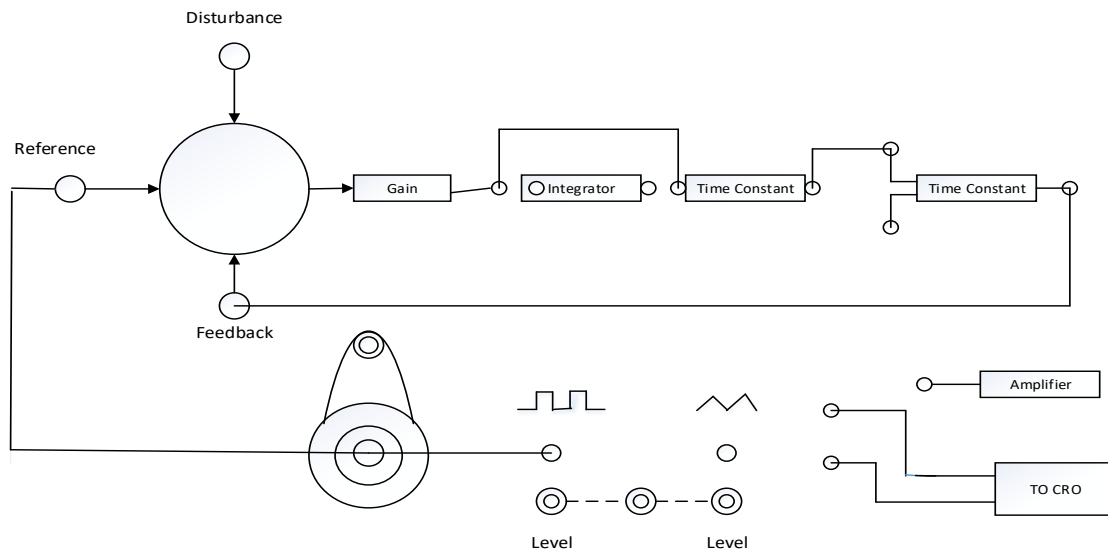


Figure 1 - Time Response of Second Order System

Try

3. Determine the time domain specifications for different zeta values for Figure 1
4. For different test signals obtain the time domain specifications for different zeta=1
5. Draw the comparison graph for first, second order system with zeta = 1, step signal.

2. TRANSFER FUNCTION OF DC MOTOR

To determine Transfer function of armature controlled dc motor 1. By performing load test on dc motor and speed control by armature voltage control. 2. To plot characteristics between back e.m.f. and angular velocity and armature current Vs torque

2.1. TRANSFER FUNCTION OF DC MOTOR with MATLAB

Determine the transfer function, time response of DC motor and verification with MATLAB

The transfer function of a DC motor can be derived from the basic equations governing its dynamics. The typical transfer function for a DC motor is given by:

$$I(s) = \frac{K}{s(Js + b)K_b^2}$$

Hint

```
% Define system parameters
% Define DC Motor Parameters
K = 0.01;    % Motor gain or torque constant (Nm/A)
J = 0.01;    % Moment of inertia (kg*m^2)
b = 0.1;     % Viscous damping coefficient (Nms/rad)
K_b = 0.01;  % Back EMF constant (V/(rad/s))
% Formulate Transfer Function
numerator = K;
denominator = [J, b, 0; 0, K_b, 0; 0, 1, 0];
sys = tf(numerator, denominator);
% Time Response Analysis
t = 0:0.01:2; % Time vector from 0 to 2 seconds with a step size of 0.01 seconds
[y, t] = step(sys, t);
```

```
% Plot the Step Response
figure
plot(t, y);
title('Step Response of DC Motor');

xlabel('Time (seconds)');
ylabel('Angular Position (rad)');

grid on;
```

Try

1. Create a Simulink model of the DC motor system. Compare the results obtained from Simulink with those obtained using the transfer function. Ensure consistency between the analytical and simulation results.
2. Use the **"bode"** function in MATLAB to analyze the frequency response of the DC motor system. Investigate how the system behaves at different frequencies.
3. Change one parameter at a time (e.g. R, L, J, b, K, Ke) and observe how it affects the step response. Explain the observed changes in terms of the motor dynamics.

2.2. TRANSFER FUNCTION OF DC MOTOR

Determine the transfer function, time response of DC motor

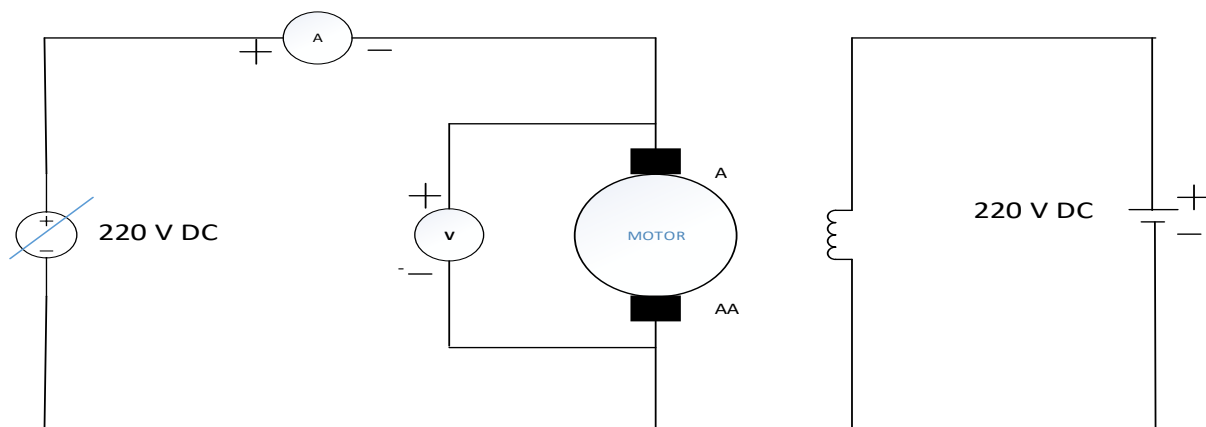


Figure 2 - Armature voltage control

Try

6. Calculate the different armature currents (i.e i_a equal i_l plus i_f) for shunt motor under various loads for figure1
7. Draw the External characteristics using armature currents (0 to 13 A) with respect to the load currents.
8. Draw the Internal characteristics using generated induced emf (0 to 220V) with respect to the field currents (0 to 2A).

3. TRANSFER FUNCTION OF AC SERVO MOTOR

The servomotors used in the low power servomechanism are a.c. servomotors. The a.c. servomotor is basically two phase induction motor. The output power of a.c. servomotor varies from fraction of watts to few hundreds of watts. The operating frequency is 50 Hz to 400 Hz.

3.1. TRANSFER FUNCTION OF AC SERVO MOTOR using MATLAB

To control an AC servo motor in MATLAB, you can use the Simulink environment along with the necessary blocks and functions. Here's a basic example of a MATLAB program for simulating the characteristics of an AC servo motor using Simulink:

```
% AC Servo Motor Characteristics Simulation Assignment
% Parameters
R = 2; % Resistance (ohms)
L = 0.01; % Inductance (henrie)
Kt = 0.1; % Torque constant (Nm/A)
J = 0.001; % Moment of inertia (kg.m^2)
B = 0.01; % Viscous friction coefficient (Nm/(rad/s))
Ke = 0.1; % Back EMF constant (V/(rad/s))
% Transfer function of the AC servo motor system
numerator = [Kt]
denominator = [J * L, (B * L + J * R), (R * B + Kt^2), 0];
motor_tf = tf(numerator, denominator);
% Create a time vector for simulation
t = 0:0.01:2;

% Simulate the step response
step_response = step(motor_tf, t);

% Plot the step response
figure;
plot(t, step_response);
title('Step Response of AC Servo Motor System');
xlabel('Time (seconds)');
ylabel('Angular Position');
grid on;

% Calculate characteristics
info = stepinfo(motor_tf);

% Display characteristics
fprintf('Overshoot: %.2f%%\n', info.Overshoot);
fprintf('Settling Time: %.2f seconds\n', info.SettlingTime);
fprintf('Rise Time: %.2f seconds\n', info.RiseTime);
```

Try

1. Consider an AC servo motor as system with the following specifications:

Find the Transfer function of Motor for given Parameters:

- a) Resistance (R): 2 ohms
- b) Inductance (L): 0.01 henries
- c) Torque constant (Kt): 0.1 Nm/A
- d) Moment of inertia (J): 0.001 kg.m²
- e) Viscous friction coefficient (B): 0.01 Nm/(rad/s)
- f) Back EMF constant (Ke): 0.1 V/(rad/s)

3.2. AC SERVO MOTOR

Determine the Torque Speed characteristics of a AC servomotor

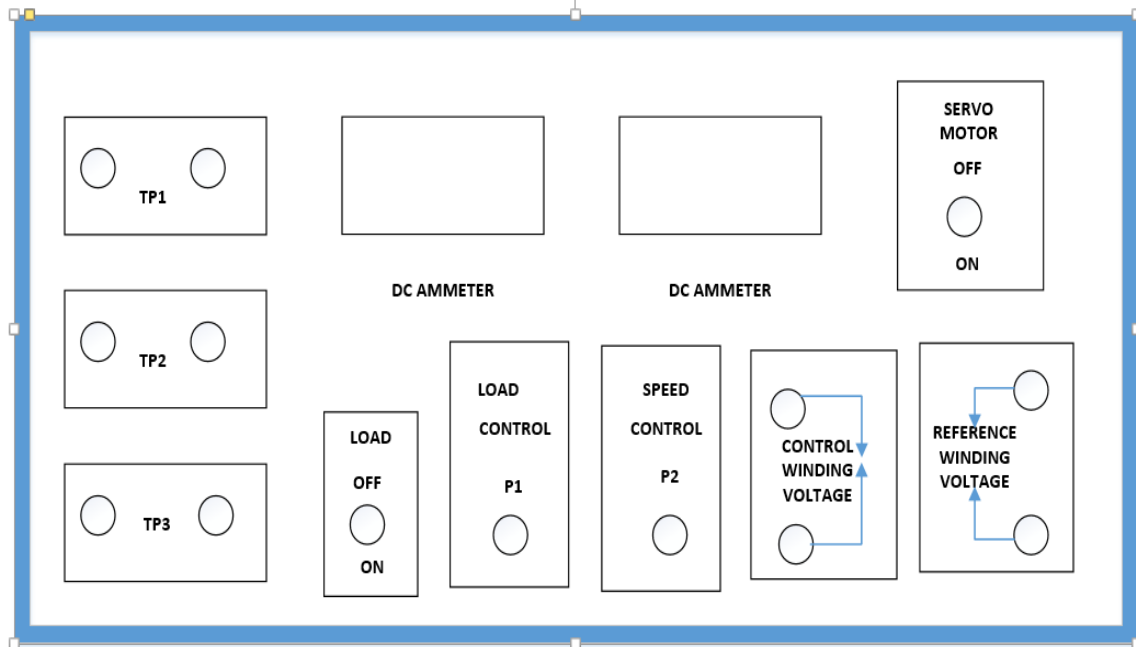


Figure 3 – Front panel of AC Servo-motor Controller

Try

1. Calculate the different armature currents (i.e I_a equal I_a plus if) for AC servo motor under various Loads for figure 3.
2. Draw the External characteristics using armature currents (0 to 13 A) with respect to the load currents.
3. Draw the Internal characteristics using generated induced emf (0 to 220V) with respect to the field currents (0 to 2A).

4. EFFECT OF VARIOUS CONTROLLERS ON SECOND ORDER SYSTEM USING MATLAB

PID controllers are adjusted on-site and many different types of tuning rules have been proposed in different literatures. Using these tuning rules, delicate and fine tuning of PID controllers can be made on-site. Also, automatic tuning methods have been developed and some of the PID controllers may possess on-line automatic tuning capabilities. Modified forms of PID control, such as PID control and two-degrees-of- freedom PID control, are currently in use in industry.

4.1. EFFECT OF VARIOUS CONTROLLERS ON SECOND ORDER SYSTEM USING MATLAB

Analyze the effects of various controllers on a second-order system using the control system toolbox. Here's a basic example to demonstrate the impact of different controllers on a second-order system:

```
% Define the second-order system
numerator = 1;
denominator = [1, 2, 1]; % Second-order system with damping ratio 1 (critically damped)
```

```

sys = tf(numerator, denominator);
% Define time vector
t = 0:0.01:5;

% Step response of the original system
figure;
step(sys, t);
title('Step Response - Original System');

% Proportional (P) controller

Kp = 1;
controller_p = tf([Kp], [1]);

% Closed-loop system with P controller

sys_p = feedback(series(controller_p, sys), 1);

% Step response of the system with P controller

figure;
step(sys_p, t);
title('Step Response - Proportional (P) Controller');
% Proportional-Integral (PI) controller
Kp = 1;
Ki = 0.5;
controller_pi = tf([Kd, Ki], [1, 0]);
% Closed-loop system with PI controller
sys_pi = feedback (series (controller_pi, sys), 1);
% Step response of the system with PI controller
figure;
step(sys_pi, t);
title('Step Response - Proportional-Integral (PI) Controller');
% Proportional-Derivative (PD) controller
Kp = 1;Kd = 0.2;
controller_pd = tf([Kp, Kd], [1]);
% Closed-loop system with PD controller

sys_pd = feedback(series(controller_pd, sys), 1);
% Step response of the system with PD controller

figure;

step(sys_pd, t);
title('Step Response - Proportional-Derivative (PD) Controller');

```

Try

1. Compare the step responses of a second-order system with Proportional (P), Proportional-Integral (PI), and Proportional-Derivative (PD) controllers.
2. Analyze and discuss the impact of each controller on system performance.

4.2. EFFECT OF VARIOUS CONTROLLERS ON SECOND ORDER SYSTEM

Determine time domain specifications of the second order system by using PID Controller

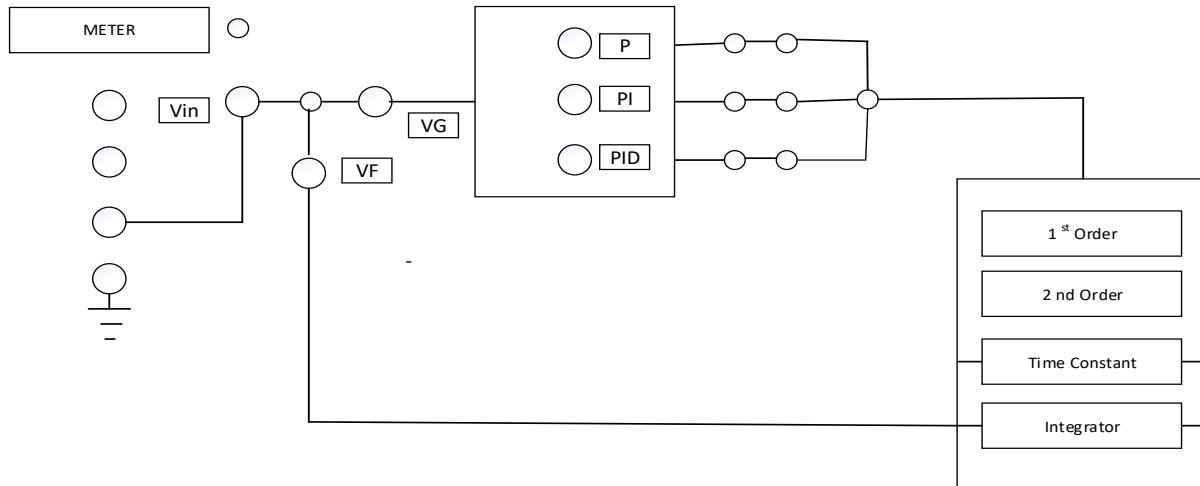


Figure 4 – Block Diagram Effect of P, PI, PID Controller on second order systems

Try

1. Choose values for natural frequency (ω_n) and damping ratio (ζ) to represent an underdamped, critically damped, and overdamped system.
2. How the proportional controller affects the system's rise time, overshoot, and steady-state error.
3. Analyze the effect of varying the proportional gain (K_p) and integral gain (K_i) on the closed-loop system.

5. COMPENSATOR

A lead compensator is commonly used for improving stability margins. Lag compensators are used to improve the steady state performance. The lead compensator achieves the desired results through the merits of its phase-lead contribution. The lag compensator accomplishes its result through the merits of its attenuation property at high frequencies

Study of Lag, Lead, Lead - Lag compensation networks and obtain its frequency response.

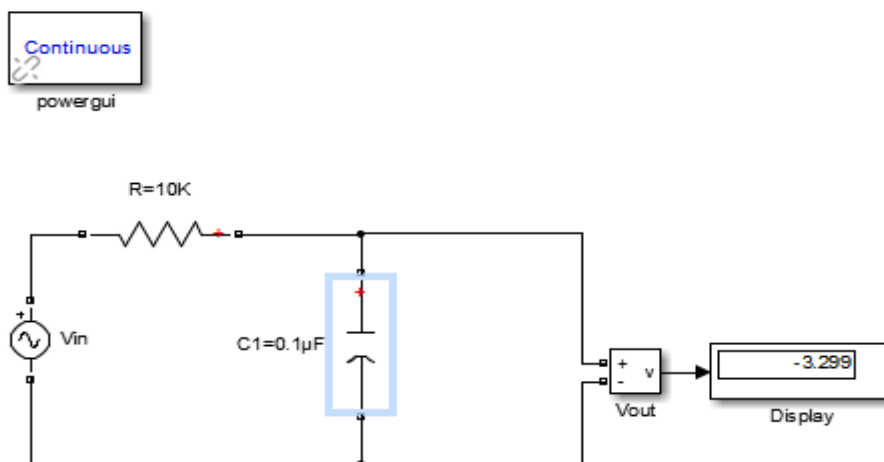


Figure 5.1 – Lag Compensation

Continuous
powergui

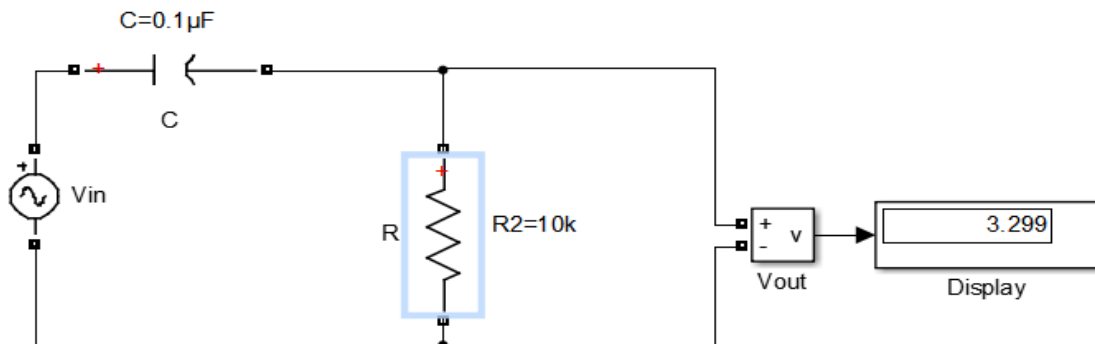


Figure 5.2 – Lead Compensation

Continuous
powergui

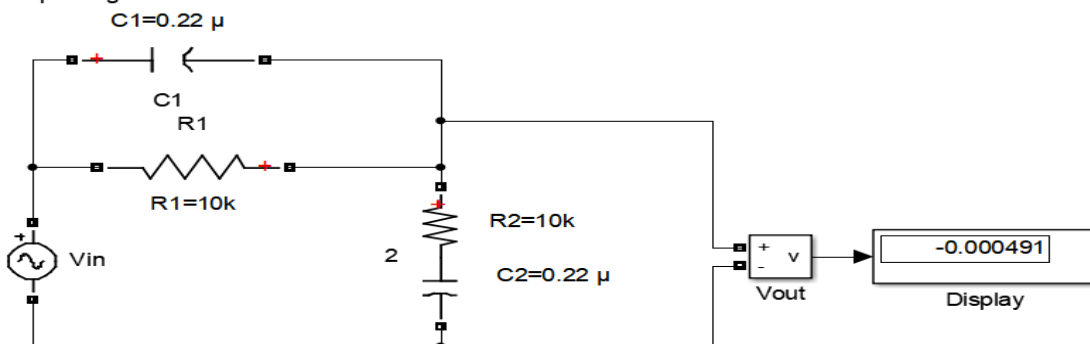


Figure 5.3 – Lag - Lead Compensation

Try

4. Consider a unity feedback control system with an open-loop transfer function:

$$G(s) = \frac{K}{s(s+3)(s+5)}$$

The system is required to meet the following specifications:

Settling Time: Less than 2 seconds.

Overshoot: Less than 10%.

Phase Margin: Greater than 40 degrees.

- Design a lead-lag compensator with parameters (K_{lead} , α , T_{lead} , K_{lag} , T_{lag}) to meet the given specifications.
- Provide explanations for your design choices and how each parameter affects the system response.
- Construct the closed-loop transfer function by connecting the lead-lag compensator to the original system.

6. TEMPERATURE CONTROLLER

PID temperature control is a loop control feature found on most process controllers to improve the accuracy of the process. PID temperature controllers work using a formula to calculate the difference between the desired temperature set point and current process temperature, then predicts how much power to use in subsequent process cycles to ensure the process temperature remains as close to the set point as possible by eliminating the impact of process environment changes.

6.1. TEMPERATURE CONTROLLER

Study the performance of PID controller used to control the temperature of an oven

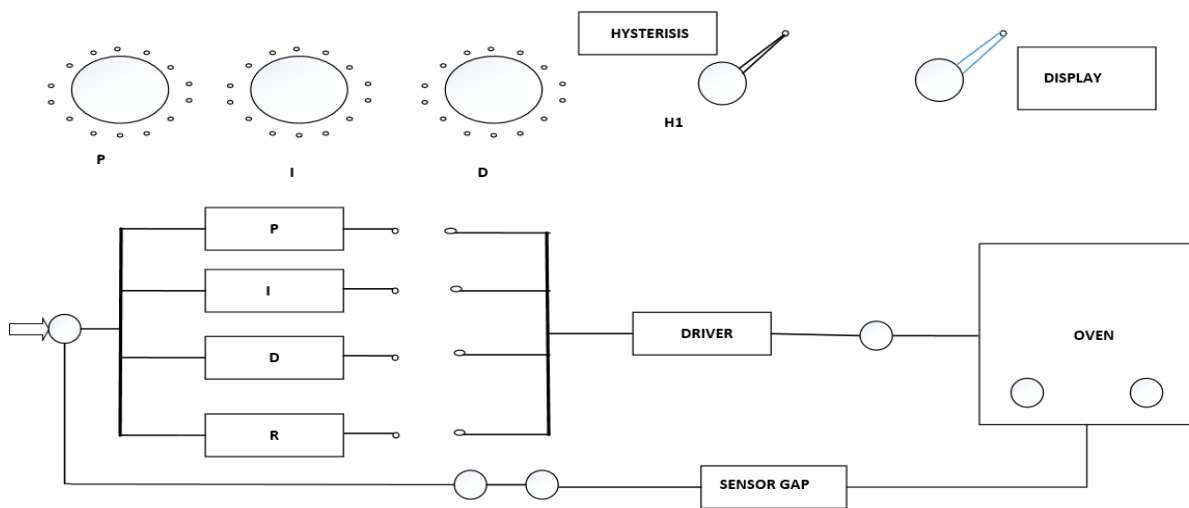


Figure 6 - Temperature controller of an oven

Try

- Design a PID controller for the oven temperature control. Experiment with different values of K_p , K_i , and K_d to achieve desired performance for figure 6
- Compare the performance of the PID controller with alternative controllers (e.g., P, PI, PD) in terms of temperature regulation for figure 6

6.2. TEMPERATURE CONTROLLER using MATLAB

To study the performance of a PID controller in controlling the temperature of an oven, you can use MATLAB and the Simulink environment. Below is a basic example with MATLAB code for simulating a PID-controlled oven temperature system and analyzing its performance:

Hint

```
% PID Controller for Oven Temperature Control

% System parameters
tau = 10; % Time constant of the oven (in seconds)

% Create a transfer function representing the open-loop system (oven)
oven_sys = tf(1, [tau, 1]);

% PID Controller parameters
Kp = 1; % Proportional gain
```

```

Ki = 0.1; % Integral gain
Kd = 0.01; % Derivative gain

% Create a PID controller
pid_controller = pid(Kp, Ki, Kd);

% Connect the PID controller to the oven system

closed_loop_sys = feedback(pid_controller * oven_sys, 1);

% Simulate the closed-loop system response to a step change in setpoint

time = 0:0.1:100; % Time vector for simulation

setpoint = 100; % Desired temperature setpoint

% Simulate the system response
[y, t] = step(setpoint * closed_loop_sys, time);

% Plot the temperature response
figure;
plot(t, y);
title('PID Controller Performance - Oven Temperature Control');
xlabel('Time (seconds)');
ylabel('Temperature');
grid on;

% Analyze the step response characteristics
step_info = stepinfo(closed_loop_sys);
disp('Step Response Characteristics:');
disp(step_info);

```

Try

1. Experiment with different tuning methods (e.g., manual tuning, Ziegler-Nichols) to optimize the PID gains.
2. Compare the system's performance before and after tuning.

7. DESIGN AND VERIFICATION OF OP-AMP BASED PID CONTROLLER

The PID control is the most commonly known for control process utilized as a part of industries for controlling action. The basic technique for PID controllers makes it simple to coordinate the process output. The main focus of the experiment is about study of OPAMP and fabrication of an PID Controller using the three control parameters. The Controller design is demonstrated through simulation in order to get an output of better dynamic and static performance. The controller is fabricated on hardware after the test of individual terms: -proportional, integral and derivative

7.1. DESIGN AND VERIFICATION OF OP-AMP BASED PID CONTROLLER USING MATLAB

Implementation of PID controller using Op-Amps and verification using MATLAB

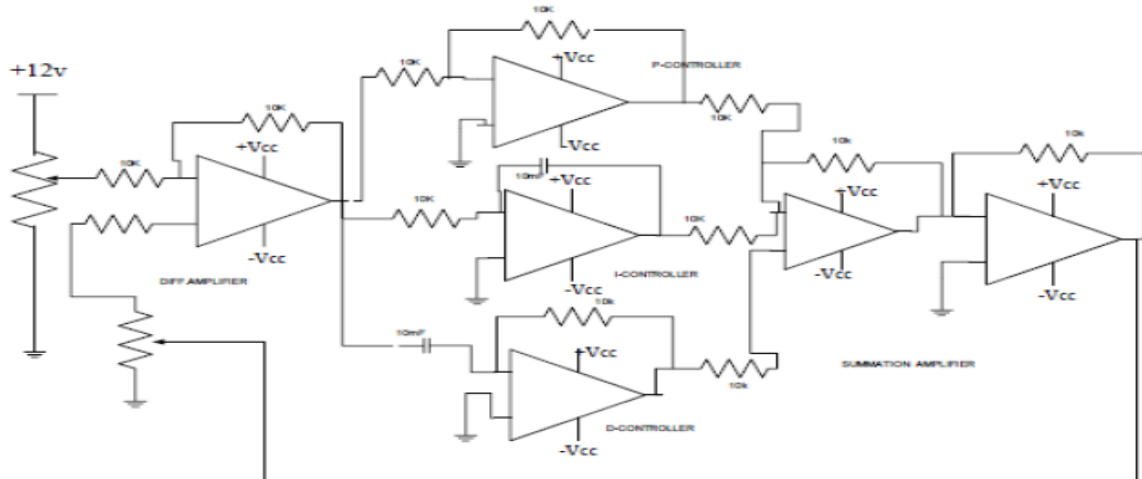


Figure 7.1 – Circuit Diagram of OP-AMP based PID Controller

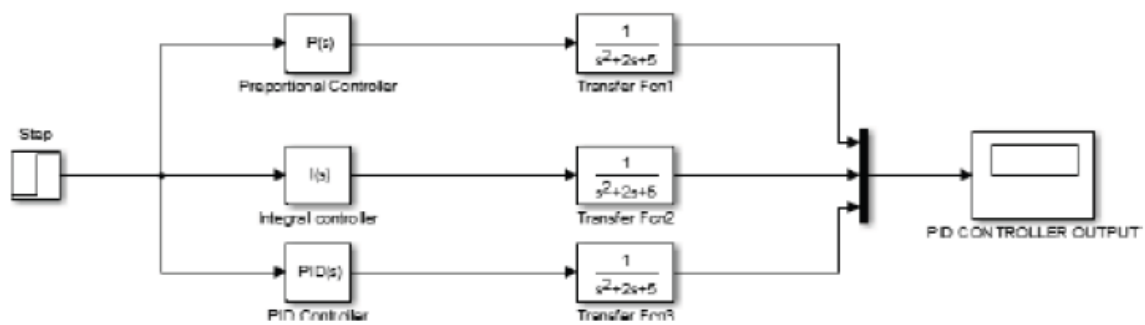


Figure 7.2 – Simulation circuit diagram of OP-AMP PID Controller

Try

3. Design an op-amp based PID controller for the chosen system.
4. Express the PID controller transfer function $H(s)$ using operational amplifiers.
5. Explain the design choices and constraints in selecting resistor and capacitor values.

7.2. DESIGN AND VERIFICATION OF OP-AMP BASED PID CONTROLLER

Implementing the design and verification of an op-amp based PID controller involves a combination of analog circuit design and simulation using tools like MATLAB. Below is an example MATLAB code to illustrate the op-amp based PID controller design and verification. Please note that this is a basic example, and you may need to adapt it according to the specifics of your system.

```
% PID Controller for Oven Temperature Control

% System parameters

% Design and Verification of Op-Amp Based PID Controller

% Plant Transfer Function (example: second-order system)
numerator = 1;
denominator = [1, 2, 1];
```

```

plant_tf = tf(numerator, denominator);

% Op-Amp Based PID Controller Parameters
R1 = 10e3; % Resistance values (adjust as needed)
R2 = 10e3;
C1 = 1e-6; % Capacitor value (adjust as needed)

% Op-Amp PID Controller Transfer Function
s = tf('s');
pid_opamp_tf = (R2 * (R1 + R2) * s + R1) / (R1 * R2 * C1 * s^2 + (R1 + R2) * s);

% Closed-Loop System Transfer Function
closed_loop_tf = feedback(pid_opamp_tf * plant_tf, 1);

% Step Response Simulation
time = 0:0.01:5;
input_signal = ones(size(time)); % Step input
output_response = lsim(closed_loop_tf, input_signal, time);

% Plot Step Response
figure;
plot(time, input_signal, 'r--', 'LineWidth', 1.5, 'DisplayName', 'Step Input');
hold on;
plot(time, output_response, 'b-', 'LineWidth', 1.5, 'DisplayName', 'Output Response');
title('Op-Amp Based PID Controller - Step Response');
xlabel('Time (seconds)');
ylabel('Amplitude');
legend('show');
grid on;

% Bode Plot of the Closed-Loop System
figure;

bode(closed_loop_tf);
title('Op-Amp Based PID Controller - Bode Plot');

% Stability Analysis
margin_info = margin(closed_loop_tf);
disp('Stability Analysis:');
disp(margin_info);

```

Try:

1. Compare the op-amp based PID controller with an ideal PID controller in terms of performance.
2. Experiment with different resistor and capacitor values to observe the effect on the closed-loop system's performance.

8. STABILITY ANALYSIS USING DIGITAL SIMULATION

stability analysis of a linear time-invariant system using root locus, Bode plot, polar plot, and Nyquist criterion through digital simulation. MATLAB can be used for this purpose.

8.1. STABILITY ANALYSIS USING ROOT LOCUS

Stability analysis using Root Locus in MATLAB.

Hint

```
% Stability Analysis using Root Locus in MATLAB

% System parameters
numerator = 1;
denominator = [1, 3, 2]; % Example second-order system: s^2 + 3s + 2
plant_tf = tf(numerator, denominator);

% Plot the root locus
figure;
rlocus(plant_tf);
title('Root Locus Analysis');
grid on;

% Customizing the plot (optional)
% set(findobj(gcf, 'Type', 'Line'), 'LineWidth', 2); % Increase line width
% xlabel('Real Part');
% ylabel('Imaginary Part');
% legend('show');

% Comment out or remove the customization lines if not needed

% Zoom into the plot (optional)
% xlim([-5, 1]);
% ylim([-5, 5]);

% Comment out or remove the xlim and ylim lines if not needed
```

Try:

1. MATLAB program for complete root locus for the system with open-loop transfer function
$$G(s) = \frac{K(S + 4)}{S(S + 2)}$$
2. MATLAB program for complete root locus for the system with open-loop transfer function
$$G(s) = \frac{K(S + 3)}{S(s^2 + S + 2)}$$

8.2. STABILITY ANALYSIS USING BODE PLOT

Stability analysis using Bode plots in MATLAB involves examining the frequency response of a system.

Hint

```
% Stability Analysis using BODE PLOT in MATLAB
Step 1: Define the Transfer Function
% Example: Second-order system
numerator = 1;
denominator = [1, 3, 2]; % s^2 + 3s + 2
system_tf = tf(numerator, denominator);
Step 2: Generate Bode Plot
% Plot Bode magnitude and phase
figure;
bode(system_tf);
```

```

title('Bode Plot Analysis');
grid on;
Step 3: Customization (Optional)
% Customize the Bode plot
set(findobj(gcf, 'Type', 'Line'), 'LineWidth', 2); % Increase line width
xlabel('Frequency (rad/s)');
ylabel('Magnitude (dB) and Phase (degrees)');
legend('show');
Step 4: Zooming and Exploration (Optional)
% Zoom into specific frequency range
xlim([0.1, 10]);
Step 5: Analytical Insights
% Obtain crossover frequency, gain margin, and phase margin
[mag, phase, wout] = bode(system_tf);
crossover_frequency = wout(find(mag <= 0, 1)); % Find the frequency where magnitude is 0 dB
gain_margin = 1/mag(find(wout <= crossover_frequency, 1)); % Gain at crossover frequency
phase_margin = 180 + phase(find(wout <= crossover_frequency, 1)); % Phase margin
Step 6: Conclusion
Summarize the stability characteristics of the system based on the Bode plot.

```

Try:

1. MATLAB program for complete BODE PLOT for the system with open-loop transfer function

$$G(S) = \frac{KS^2}{(1+S)(1+0.1S)(1+0.01S)}$$

2. MATLAB program for complete BODE PLOT for the system with open-loop transfer function

$$G(S) = \frac{Ke^{-0.2S}}{(1+S)(1+0.1S)(1+0.2S)}$$

8.3. STABILITY ANALYSIS USING POLAR PLOT

Stability analysis using polar plots in MATLAB involves examining the polar representation of the system's frequency response. Below are the MATLAB steps for generating a polar plot and performing stability analysis:

Hint

```

Step 1: Define the Transfer Function
% Example: Second-order system
numerator = 1;
denominator = [1, 3, 2]; % s^2 + 3s + 2
system_tf = tf(numerator, denominator);
Step 2: Generate Polar Plot
% Plot polar representation
figure;

polarplot(angle(system_tf), abs(system_tf));
title('Polar Plot Analysis');
grid on;
Step 3: Customization (Optional)
% Customize the polar plot
ax = gca;
ax.ThetaZeroLocation = 'top';
ax.ThetaDir = 'clockwise';

Step 4: Analytical Insights

```


% Analyze polar plot to understand system stability

Step 5: Conclusion

Summarize the stability characteristics of the system based on the polar plot.

Try:

1. MATLAB program for complete POLAR PLOT for the system with open-loop transfer function

$$G(S) = \frac{1}{(1+S)(1+4S)}$$

2. MATLAB program for complete POLAR PLOT for the system with open-loop transfer function

$$G(S) = \frac{(1+0.1S)(1+0.2S)}{S^3(1+0.004S)(1+0.001S)}$$

8.4. STABILITY ANALYSIS USING NYQUIST PLOT

Stability analysis using NYQUIST PLOTS in MATLAB involves examining the frequency response of a system.

Hint

Step 1: Define the Transfer Function

% Example: Second-order system

numerator = 1;

denominator = [1, 3, 2]; % $s^2 + 3s + 2$

system_tf = tf(numerator, denominator);

Step 2: Generate Nyquist Plot

% Plot Nyquist plot

figure;

nyquist(system_tf);

title('Nyquist Plot Analysis');

grid on;

Step 3: Customization (Optional)

% Customize the Nyquist plot

set(findobj(gcf, 'Type', 'Line'), 'LineWidth', 2); % Increase line width

xlabel('Real Part');

ylabel('Imaginary Part');

legend('show');

Step 4: Analytical Insights

% Analyze Nyquist plot to understand system stability

Step 5: Conclusion

Summarize the stability characteristics of the system based on the Nyquist plot.

Try:

1. MATLAB program for complete NYQUIST PLOT for the system with open-loop transfer function

$$G(S) = \frac{1}{(1+S)(1+4S)}$$

2. MATLAB program for complete NYQUIST PLOT for the system with open-loop transfer function

$$G(S) = \frac{1}{S(1+S)(1+4S)}$$

9. STATE SPACE MODEL USING DIGITAL SIMULATION

Verify a state-space model from a transfer function and vice versa using digital simulation in MATLAB.

1.Verification of State-Space Model from Transfer Function:

% Define the transfer function

```

numerator = [1];
denominator = [1, 2, 1];
sys_tf = tf(numerator, denominator);

% Convert transfer function to state-space representation
sys_ss = tf2ss(sys_tf);

% Display state-space matrices
disp('State-Space Matrices:');
disp('A matrix:');
disp(sys_ss.A);
disp('B matrix:');
disp(sys_ss.B);
disp('C matrix:');
disp(sys_ss.C);
disp('D matrix:');
disp(sys_ss.D);
2. Verification of Transfer Function from State-Space Model:
% Define state-space matrices
A = [-2 -1; 1 0];
B = [1; 0];
C = [1 0];
D = 0;
% Convert state-space model to transfer function
sys_tf_verified = ss2tf(A, B, C, D);

% Display transfer function coefficients
numerator_verified = sys_tf_verified.Numerator{1};
denominator_verified = sys_tf_verified.Denominator{1};
disp('Transfer Function Coefficients:');
disp('Numerator Coefficients:');
disp(numerator_verified);
disp('Denominator Coefficients:');
disp(denominator_verified);
3.Digital Simulation for Verification:
% Define time vector
t = 0:0.01:5;
% Simulate the transfer function response
y_tf = lsim(sys_tf, ones(size(t)), t);
% Simulate the state-space model response
u = ones(size(t));
x0 = [0; 0]; % Initial state
[y_ss, t_ss, x_ss] = lsim(sys_ss, u, t, x0);

% Plot results for verification
figure;
subplot(2,1,1);
plot(t, y_tf, 'r-', 'LineWidth', 1.5, 'DisplayName', 'Transfer Function');
hold on;
plot(t_ss, y_ss, 'b--', 'LineWidth', 1.5, 'DisplayName', 'State-Space Model');
title('Output Response');
xlabel('Time');
ylabel('Amplitude');
legend('show');

```

```
subplot(2,1,2);
plot(t_ss, x_ss(:,1), 'b-', 'LineWidth', 1.5, 'DisplayName', 'State 1');
hold on;
plot(t_ss, x_ss(:,2), 'g-', 'LineWidth', 1.5, 'DisplayName', 'State 2');
title('State Variables');
xlabel('Time');
ylabel('Amplitude');
legend('show');
```

Try

3. MATLAB Program to obtain the state model of the system

$$T(S) = \frac{Y(S)}{U(S)} = \frac{2S^3 + 7S^2 + 12S + 8}{S^3 + 6S^2 + 11S + 9}$$

4. MATLAB program to obtain the transfer function for the state model

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -9 & -11 & -6 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, C = [-10 \quad -10 \quad -5]$$

9.1 Introduction to PLC and Industrial Automation

A Programmable Logic Controller (PLC) is a specialized digital computer used in industrial automation and control systems. It plays a crucial role in automating processes in manufacturing plants, power generation facilities, water treatment plants, and various other industrial applications. PLCs are designed to monitor inputs, make decisions based on a programmed logic, and control outputs to automate machinery and processes.

Basic introduction to a Programmable Logic Controller (PLC) and an industrial automation laboratory:

Programmable Logic Controller (PLC):

a. Purpose:

- PLCs are used for automating and controlling industrial processes.
- They replace traditional relay-based control systems with a more flexible and efficient digital control.

b. Components:

- CPU (Central Processing Unit): The brain of the PLC, responsible for executing the control program.
- Input Modules: Receive signals from sensors and other devices.
- Output Modules: Control actuators, motors, and other output devices.
- Memory: Stores the control program and data.
- Facilitate communication with other devices and systems.

c. Programming:

- PLCs are programmed using specialized software that often employs ladder logic diagrams, resembling relay logic.
- Programming involves defining inputs, outputs, and the logic that determines the system's behavior.

d. Applications:

- PLCs are used in various industries, including manufacturing, chemical processing, power plants, and more.

- They control processes such as assembly lines, batch processing, and continuous production.

Industrial Automation Laboratory:

An industrial automation laboratory is a controlled environment where students, engineers, or technicians can learn and experiment with various aspects of industrial automation using PLCs. Such a lab typically includes:

a. Hardware Setup:

- PLCs and associated input/output modules.
- Sensors and actuators representing real-world industrial equipment.
- Communication interfaces for networking and data exchange.

b. Programming Tools:

- Computers with PLC programming software.
- Simulation tools to test programs before deploying them to actual PLCs.

c. Training Stations:

- Workstations equipped with PLCs, input/output devices, and connection terminals for hands-on learning.

d. Experiments and Projects:

- Laboratory sessions designed for students to implement control strategies, troubleshoot, and optimize industrial processes.
- Projects that involve designing and implementing automation solutions.

e. Safety Measures:

- Protocols to ensure safe experimentation, especially when dealing with real-world industrial equipment.

f. Documentation and Analysis:

- Tools and methodologies for documenting PLC programs, analyzing system performance, and identifying areas for improvement.

Try:

PLC Ladder Diagrams:

- Analyzing the results of operations of timers, counters, DC/AC motors and traffic signal control using PLC ladder diagrams in terms of concepts introduced in the Industrial Automation and Control course.

Hardware Connections:

- Trained to get hands-on experience on wiring the Delta make DVP series Programmable Logic Controller and Human Machine Interface.

10. LADDER DIAGRAMS USING PLC

Input output connection, simple programming, ladder diagrams, uploading, running the program and debugging in

1. Input-Output Connection:

PLCs interface with the external world through input and output connections. Inputs can be connected to sensors, switches, or other devices, while outputs can be connected to actuators, motors, or other controlled devices. These connections are typically made through terminals on the PLC.

2. Simple Programming using Ladder Diagrams:

Ladder Logic is a graphical programming language widely used in PLCs. Below is an example of a simple start-stop motor control program using ladder diagrams:

```
--[ Start ]----( )----[ Motor Coil ]--  
                |  
--[ Stop ]-----[ ]-----
```

In this example, the motor will start when the "Start" button is pressed (closed), and it will stop when the "Stop" button is pressed (closed).

3. Uploading a Program to PLC:

Uploading involves transferring the ladder logic program from your development environment (PLC programming software) to the PLC. This is typically done through a communication cable. The uploaded program is stored in the PLC's memory.

4. Running the Program:

Once the program is uploaded, the PLC can be set to "Run" mode. In this mode, it executes the ladder logic continuously based on the current state of input devices. Outputs change state accordingly.

5. Debugging:

Debugging in PLC programming involves identifying and correcting errors in the ladder logic or addressing. Common debugging tools include:

Online Monitoring: Observe the current status of inputs and outputs in real-time.

Force I/O: Force inputs or outputs to specific states for testing.

Setting Breakpoints: Pause execution at specific points to inspect variables or conditions

Example: Emergency Stop Circuit

```
--[ Start ]----( )----[ Motor Coil ]----  
                |  
--[ E-Stop ]-----[ ]-----
```

In this example, the motor will start when the "Start" button is pressed, but it will immediately stop if the emergency stop ("E-Stop") button is pressed.

Try:

Design a ladder diagram using a PLC to control a conveyor belt system

11. TRUTH TABLES USING PLC

In programmable logic controllers (PLCs), truth tables are often used to describe the relationship between inputs and outputs in a logic circuit. Each row in the truth table represents a combination of input states and the corresponding output state. Here is an example of a truth table using ladder logic for a basic AND gate:

Example: AND Gate Truth Table and Ladder Logic

Consider a simple AND gate with two inputs (I1 and I2) and one output (O).

Truth Table:

I1	I2	O
0	0	0
0	1	0
1	0	0
1	1	1

```
|----[ I1 ]-----[ ]----[ O ]--|  
|                               |  
|----[ I2 ]-----|
```

In this example:

1. If both I1 and I2 are 0, the output O is 0.
2. If either I1 or I2 is 1 (but not both), the output O is 0.
3. If both I1 and I2 are 1, the output O is 1.

Steps for Creating a Truth Table using PLC:

Identify Inputs and Outputs:

Determine the number of inputs and outputs in your logic circuit.

List All Input Combinations:

Enumerate all possible combinations of input states. This creates the input columns of your truth table.

Determine Output for Each Combination:

For each combination of input states, determine the corresponding output state.

Create Truth Table:

Organize the input and output states into a table format.

Implement Ladder Logic:

Create ladder logic based on the truth table. Use symbols such as contacts (representing inputs) and coils (representing outputs).

Try

1. Draw the OR, NAND, NOT, XNOR LOGIC gates.

12. IMPLEMENTATION OF COUNTER

In programmable logic controllers (PLCs), counters are often used to count events or pulses. There are different types of counters, such as up counters, down counters, and up/down counters. Here, I'll demonstrate the implementation of a basic up counter using ladder logic in a PLC.

Example: Up Counter Implementation

Consider an up counter with one input (Start) and one output (Count). The counter increments each time the Start input is activated.

Ladder Logic:

```
|----[ Start ]----[ ]----( )----[ Count ]--|
```

|

In this ladder logic diagram:

[Start]: Represents the input contact. When this input is closed (activated), the counter will increment.

[]: Represents the coil. This is the output that changes state when the counter increments.

(): Represents the counter. Each time the Start input is activated, the counter increments by 1, and the Count output turns on.

Steps for Implementing an Up Counter:

Identify Inputs and Outputs:

Determine the input (e.g., a Start button) and output (e.g., Count) for your up counter.

Create Ladder Logic Diagram:

Use ladder logic symbols to represent the input contact, the counter, and the output coil. Connect them appropriately to form the ladder diagram.

Configure Counter Parameters:

In your PLC programming software, configure the counter parameters such as the preset value (the value at which the counter resets to zero) and the counting direction (up, down, or up/down).

Program the Counter Logic:

Write the ladder logic program based on the ladder diagram. Define the conditions for the counter to increment and reset based on the input.

Upload and Run the Program:

Upload the ladder logic program to the PLC. Set the PLC to "Run" mode to execute the program.

Test and Debug:

Test the counter by activating the Start input and observing the Count output. Use debugging tools provided by your PLC programming software to identify and fix any issues.

Try

1. Write the ladder diagram for modifying the Up Counter for Reset counter.

13. BLINKING LIGHTS USING PLC

Implementing blinking lights using a Programmable Logic Controller (PLC) involves creating a simple ladder logic program. Below is an example of ladder logic for a PLC program that controls two blinking lights alternately.

Consider two lights (L1 and L2) that should blink alternately. The program will turn on one light while turning off the other and vice versa.

```
|----[ Timer1 ]----[ Coil_L1 ]----|
|                               |
|----[ Timer2 ]----[ Coil_L2 ]----|
```

In this ladder logic diagram:

- **[Timer1]:** Represents the first timer. When this timer reaches its preset time, it activates the associated output, Coil_L1.
- **[Coil_L1]:** Represents the output coil for Light 1. It turns Light 1 on when activated.
- **[Timer2]:** Represents the second timer. When this timer reaches its preset time, it activates the associated output, Coil_L2.
- **[Coil_L2]:** Represents the output coil for Light 2. It turns Light 2 on when activated.

Steps for Implementing Blinking Lights:**Identify Outputs:**

Determine the outputs (lights) that you want to control. In this example, we have Light 1 (L1) and Light 2 (L2).

Configure Timers:

Configure two timers (Timer1 and Timer2) in your PLC programming software. Set the preset time for each timer according to the desired blink rate.

Create Ladder Logic:

Use ladder logic symbols to represent the timers and output coils. Connect them appropriately to create the ladder diagram.

Program the Logic:

Write the ladder logic program based on the ladder diagram. Define the conditions for the timers to activate the output coils.

Upload and Run the Program:

Upload the ladder logic program to the PLC. Set the PLC to "Run" mode to execute the program.

Observe Blinking Lights:

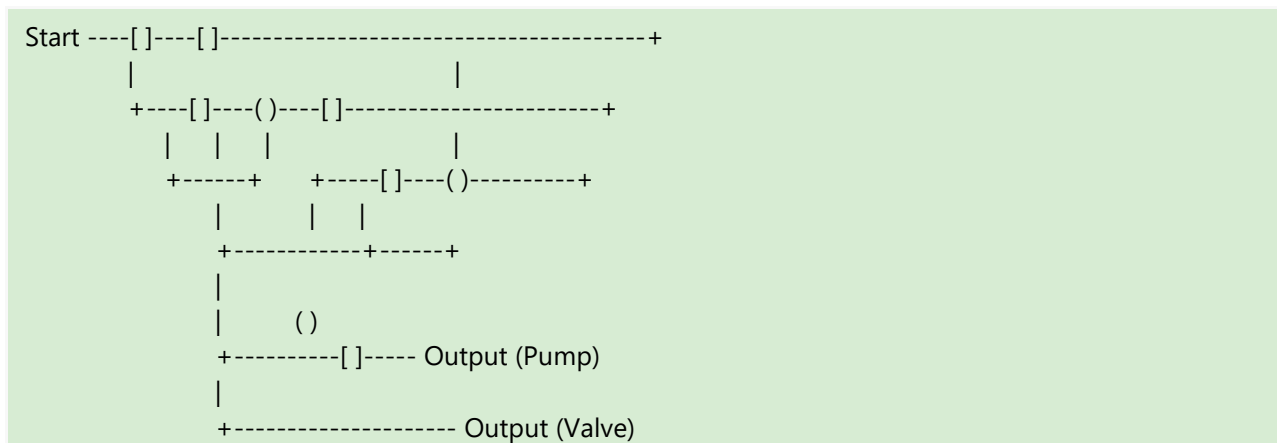
Observe the lights connected to Coil_L1 and Coil_L2 alternately blinking based on the timer settings.

Try

1. Write the ladder diagram for modifying for Additional Lights

14. WATER-LEVEL CONTROL USING PLC

Let's consider a basic scenario where you have a water tank, a pump to fill the tank, and a valve to drain the tank. We'll use two float switches: one for the low water level (LW) and one for the high water level (HW).



Explanation:

Start: Represents the start of the ladder logic program.

[]: Normally open contact (input condition).

(): Coil (output or action).

Ladder Logic Steps:

1. When the start condition is true, check if the low water level float switch ([]) is open.
2. If the low water level float switch is open, activate the pump (Pump []).
3. Check if the high water level float switch ([]) is open.
4. If the high water level float switch is open, activate the valve (Valve []).
5. Continue monitoring and repeating the logic.

Try:

1. Explain different types of sensors used for water-level measurement in tanks.
2. Write a ladder logic program for water-level control using a PLC. Include conditions for starting and stopping a pump and a valve based on low and high water-level thresholds.

V. REFERENCE BOOKS:

1. J Nagrath, M Gopal, "Control Systems Engineering", New Age International, 3rd edition, 2007.
2. K Ogata, "Modern Control Engineering", Prentice Hall, 4th edition, 2003.
3. Benjamin Kuo, "Automatic Control Systems", PHI, 7th edition, 1987.

VI. ELECGRONIC RESOURCES:

1. <https://www.ee.iitkgp.ac.in>
2. <https://www.iare.ac.in/>

VII. MATERIALS ONLINE

1. Course template
2. Lab Manual



INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal - 500 043, Hyderabad, Telangana

COURSE CONTENT

ANALOG AND DIGITAL CIRCUITS LABORATORY								
IV Semester: EEE								
Course Code	Category	Hours/Week			Credits	Maximum Marks		
AECD17	Core	L	T	P	C	CIA	SEE	Total
		0	0	2	1	40	60	100
Contact Classes: Nil	Tutorial Classes: Nil	Practical Classes: 45			Total Classes: 45			
Pre-requisite: Electronic Devices And Circuits Laboratory Digital System Design Laboratory.								

I. COURSE OVERVIEW:

The objective of this laboratory course is to meet the requirements of practical work meant for components basics, analysis and design and provides hands-on experience by examining the characteristics of various semiconductor devices and measuring instruments. This lab covers the analysis of the characteristics of semiconductor devices and functionality of the digital circuits to use as elementary blocks in analog and digital circuit applications. Students will proficiency with the capability to use simulation tools for performing various analysis of semiconductor devices, combinational and sequential circuit applications.

II. COURSE OBJECTIVES:

The students will try to learn:

- I The characteristics and applications of diodes.
- II The characteristics of transistor in different configurations.
- III The function and applications of gates.
- IV The different combinational circuits.

III. COURSE OUTCOMES:

After successful completion of the course, students should be able to:

- CO1 Apply the PN junction characteristics for the diode applications such as half wave and full wave rectifier.
- CO2 Apply the volt-ampere characteristics of PN junction diode for finding cut-in voltage, static and dynamic resistance.
- CO3 Analyze the input and output characteristics of transistor configurations for determining the input-output resistances.
- CO4 Identify the functionality of Boolean expressions using gates such as and, or, not, nand, nor, xor and Xnor.
- CO5 Build combinational circuits such as adder, subtractor, multiplexers and comparators realization using low level elementary blocks.
- CO6 Construct shift registers using the functionality of the flip flops.

DO's

1. Once the operation is completed pull the plug itself rather chord attached to it.
2. To repair the equipment switch-off the supply and go on.
3. To operate the equipment on supply, see that hands are dry, if that is not possibly hide the hand in the pockets.
4. If a person comes in contact with current unexpectedly don't touch the person with hands but immediately use any insulator material and shut down the power
5. If water is nozzles on the equipment, immediately shunt down the power using circuit breaker or pull out the plug.
5. Use the connecting wires of good continuity, short circuit of connecting wire leads damage of circuit parameters

DON'Ts

1. Do not wear loose clothing and do not hold any conducting materials in contact with skin when the power is on.
2. Do not pull out the connections until unless all the currents are dead.
3. Do not wait for switches to de-magnetize when there is a delay but pull out the plug.
4. Do not overload the circuit by plugging in too many appliances.
5. If you are mentally and physically stressed don't operate the power equipment.
6. Never operate the equipment under wet conditions.
7. Do not interconnect two or more wires, take appropriate length of wire.

SAFETY NORMS

1. The lab must be equipped with fire extinguisher.
2. See that the connections are made tight.
3. Use single plug for each equipment.
4. Cover the body completely to avoid arc effect.
5. To change the connections during the experiment, switch off the supply and carry on.
6. Used equipment may get heated, so take care handling the equipment after it is used.
7. Do the wiring, all set ups and check the circuit connections before the supply is on

IV. COURSE CONTENT:

EXERCISES FOR ANALOG AND DIGITAL CIRCUITS LABORATORY

Note: Students are encouraged to bring their own laptops for laboratory practice session

Introduction

This laboratory course enables students to get practical experience on analog and digital electronics circuits. Analog components and circuits like p-n junction diode, OP-AMP etc. and digital electronics exercises like logic gates, adders and subtractors, flip flops, counters etc. Simulation packages preferred are vivado software, Multisim etc. Analog Electronics is one of the fundamental courses found in all Electrical Engineering and most science programs. Analog circuit's process signals with continuous variation of voltage. The different Components that are normally used in Analog Electronics are:

1. Bi polar Junction Transistors
2. MOSFET's
3. OP-AMP

1. Getting Started Exercises

The semi-conductor diode is created by simply joining an n-type and a p-type material together nothing more just the joining of one material with a majority carrier of electrons to one with a majority carrier of holes.

1.1 Determine the V-I characteristics of forward Bias and Reverse Bias P-N junction diode as shown in fig 1.1.1 and 1.1.2 and draw the V-I Characteristics.

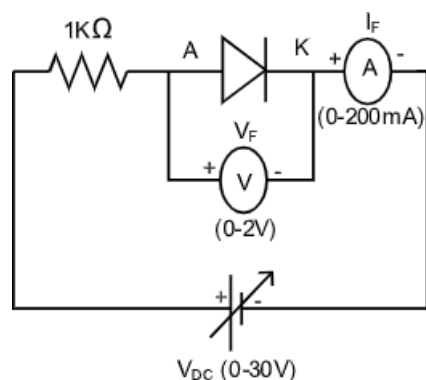


Fig 1.1.1. Forward Bias P-N Junction diode

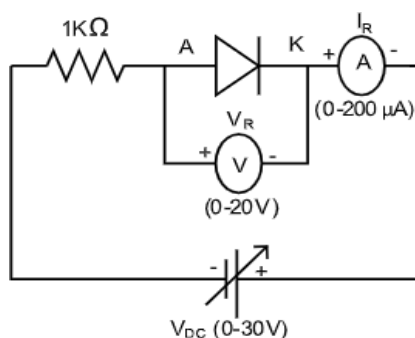


Fig 1.1.2. Reverse Bias P-N Junction diode

1.2 A diode connected to an external resistance and an e.m.f. assuming that the barrier potential developed in diode is 0.5V as shown in the fig1.3. Obtain the value of current in the circuit in milli- ampere as shown in the fig.1.2.

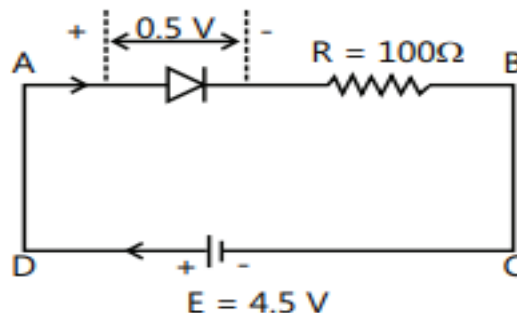


Fig 1.2. Barrier potential in diode

1.3 A diode has a constant voltage drop of 0.5 V at all current and a maximum power rating of 100 mill watts. What should be the value of the resistor R, connected in series with the diode for obtaining maximum current?

Try

- 1.3.01 Plot the V-I Characteristics of germanium diode and find the cut in voltage.
- 1.3.02 Using ua 741 Opamp, design a 1 kHz Relaxation Oscillator with 50% duty cycle. And simulate the same.

2. Exercises on Transistor Common Emitter Characteristics

Bipolar junction transistor (BJT) is a 3 terminal (emitter, base, collector) semiconductor device. There are two types of transistors namely NPN and PNP. It consists of two P-N junctions namely emitter junction and collector junction. In Common Emitter configuration the input is applied between base and emitter and the output is taken from collector and emitter. Here emitter is common to both input and output and hence the name common emitter configuration.

2.1 Verification of input and output characteristics of CE Configuration using hardware shown in fig 2.1 and draw the Waveforms.

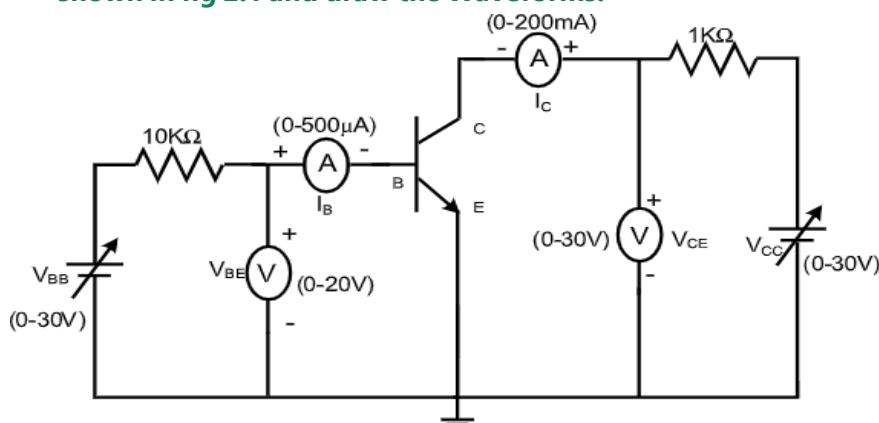


Fig 2.1. CE configuration

Try

- 2.1.01 Determine the Gain and Bandwidth of CE (Common Emitter) amplifier using hardware.
- 2.1.02 Obtain Drain characteristics and Transfer characteristics of FET.

3. Exercises on Transistor Common Base Characteristics

Bipolar junction transistor (BJT) is a 3 terminal (emitter, base, collector) semiconductor device. There are two types of transistors namely NPN and PNP. It consists of two P-N junctions namely emitter junction and collector junction. In Common base configuration the input is applied between base and emitter and the output is taken from collector and base. Here base is common to both input and output and hence the name common base configuration.

3.1 Verification of input and output characteristics of CB Configuration using hardware shown in Fig 3.1 and draw the Waveforms

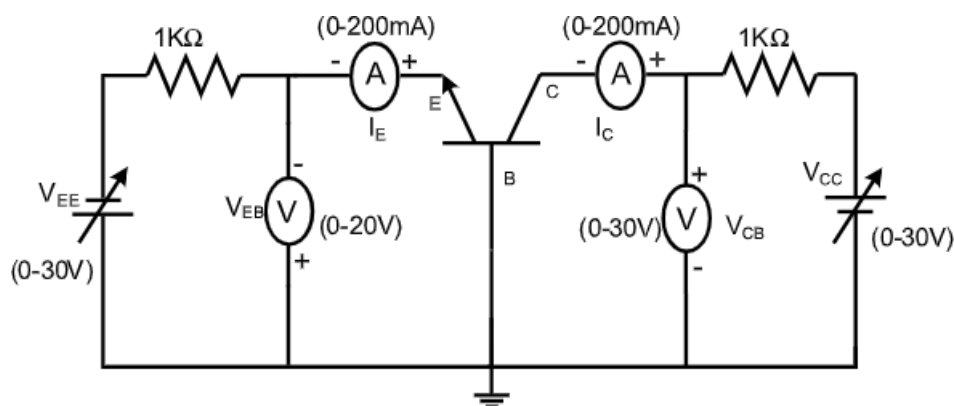


Fig 3.1. Circuit Diagram for transistor as common base configuration

Try

- 3.1.01 Determine the Gain and Bandwidth of CB (Common Base) amplifier using hardware.
- 3.1.02 Obtain Drain characteristics and Transfer characteristics of FET.

4.Exercises on Half wave rectifier and Full wave rectifier

Half-wave rectifier is an electronic circuit that converts only one-half of the AC cycle into pulsating DC. It uses only half of the AC cycle for the conversion process. Full-wave rectifier is an electronic circuit that converts the entire cycle of AC into pulsating DC. It has two diodes, and its output uses both halves of the AC signal. During the period that one diode blocks the current flow, the other diode conducts and allows the current. Both half-wave and full-wave rectifiers have their merits and demerits.

4.1 Verification of Half wave rectifier without filters using hardware shown in fig 4. 1 and draw the waveforms.

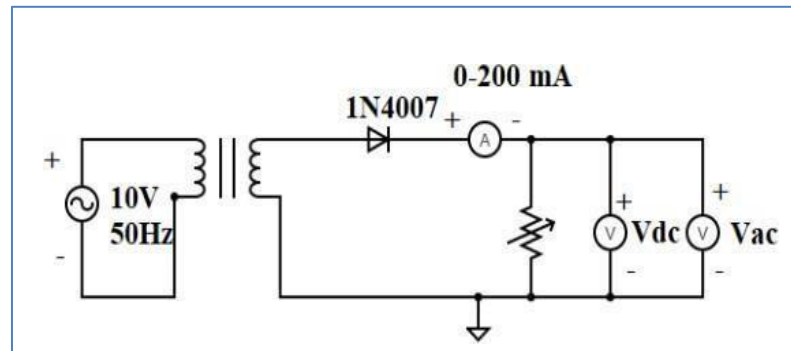


Fig 4.1. Half wave rectifier without filter

4.2 Verification of Half wave rectifier with filters using hardware shown in fig 4.2 and draw the waveforms.

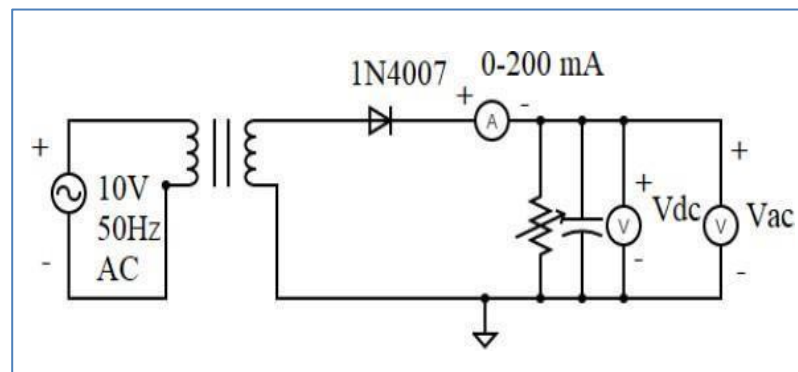


Fig 4.2. Half wave rectifier with filter

4.3 Verification of Full wave rectifier without filters using hardware shown in fig 4.3 and draw the waveforms.

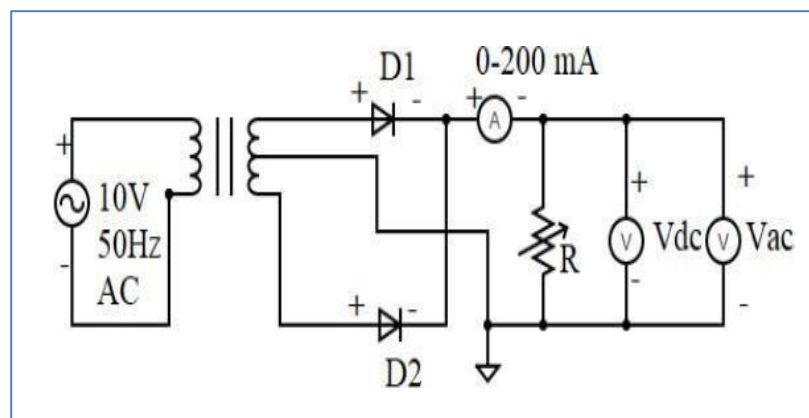


Fig 4.3 Full wave rectifier without filter

4.4 Verification of Full wave rectifier with filters using hardware shown in fig 4.4 and draw the waveforms.

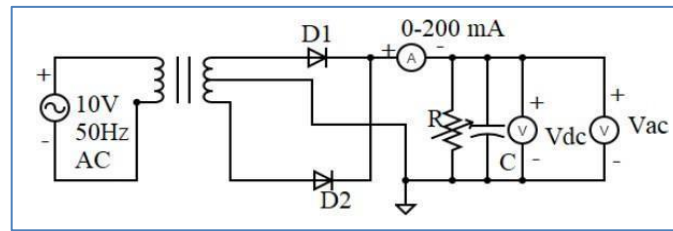


Fig 4.4 Full wave rectifier with filter

Try

- 4.4.1 Design and verify bridge rectifier with and without filter.
- 4.4.2 Design and verify precision rectifier.

5.Exercises on Drain and Transfer Characteristics FET

The functioning of Junction Field Effect Transistor depends upon the flow of majority carriers (electrons or holes) only. Basically, JFETs consist of an n type or p-type silicon bar containing p-n junctions at the sides. To study the drain and transfer characteristics of FET and find the drain resistance, trans-conductance and amplification factor for the Figure 5.1

Try

1. Plot the drain and transfer characteristics of P-channel JFET BFW11/10/BF245A with $R_L = 50 \Omega$.
2. A JFET has the following parameters: $I_{DSS} = 32 \text{ mA}$; $V_{GS}(\text{off}) = -8\text{V}$; $V_{GS} = -4.5\text{V}$. Find the value of drain current.
3. An N-channel JFET having $V_p = -4\text{V}$ and $V_{DSS} = 10\text{mA}$ is used in the circuit of Figure 5.1.1. The parameter values are $V_{DD} = 18\text{V}$, $R_s = 2\text{k}\Omega$, $R_1 = 450\text{k}\Omega$ and $R_2 = 90\text{k}\Omega$. Determine I_D and V_{DS} .

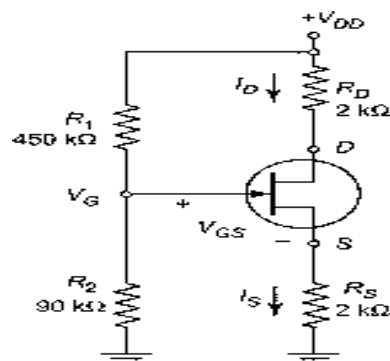


Figure 5.1.1

6.Exercises on RC Phase Shift Oscillator

RC phase shift oscillator has a CE amplifier followed by three sections of RC phase shift feedback networks. The output of the last stage is return to the input of the amplifier. The values of R and C are chosen such that the phase shift of each RC section is 60°. Thus, the RC ladder network produces a total phase shift of 180° between its input and output voltage for the given frequencies. Since CE amplifier produces 180° phase shift the total phase shift from the base of the transistor around the circuit and back to the transistor will be exactly 360°. To explore and understand oscillator design and need to perform the following practical exercises.

6.1 Design an RC phase oscillator with Resistance $R_1=R_C=10\text{K}\Omega$, $R_E=1\text{ K}\Omega$, $R=10\text{ K}\Omega$, $C_E=C_C=1\mu\text{F}$ and $C=10\mu\text{F}$ for the circuit shown in figure.6.1 and determine the frequency of oscillation of the circuit.

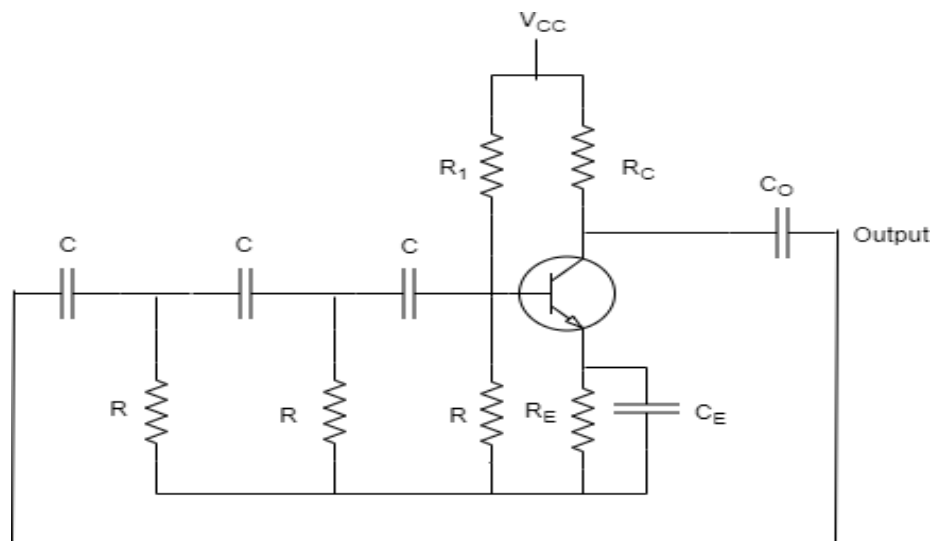


Figure 6.1: RC Phase shift oscillator

Try

1. Design an RC phase shift oscillator to produce an oscillation frequency of 2 K Hz and compare the theoretical and practical frequencies.
2. Design an RC phase shift oscillator with a target frequency of 1.5 K Hz and an amplitude of 5 Volts peak-to-peak.
3. Determine the individual and total phase shift introduced by each stage for a given RC phase shift oscillator for a frequency of 3.5 K Hz.

7.Inverting and Non-Inverting Operational Amplifiers

Inverting amplifiers are used in a number of applications like phase shifter, integration, signal balancing, mixer circuits, etc. Non-inverting amplifiers are used in circuits where high input impedance is required. They are used as voltage followers, isolation of cascaded circuits, to perform mathematical simulations.

7.1 Design an inverting amplifier circuit with OP AMP 741C for gain of 10.

Plot the waveforms, observe the phase reversal, measure the gain for the circuit shown in Figure.7.1.1.

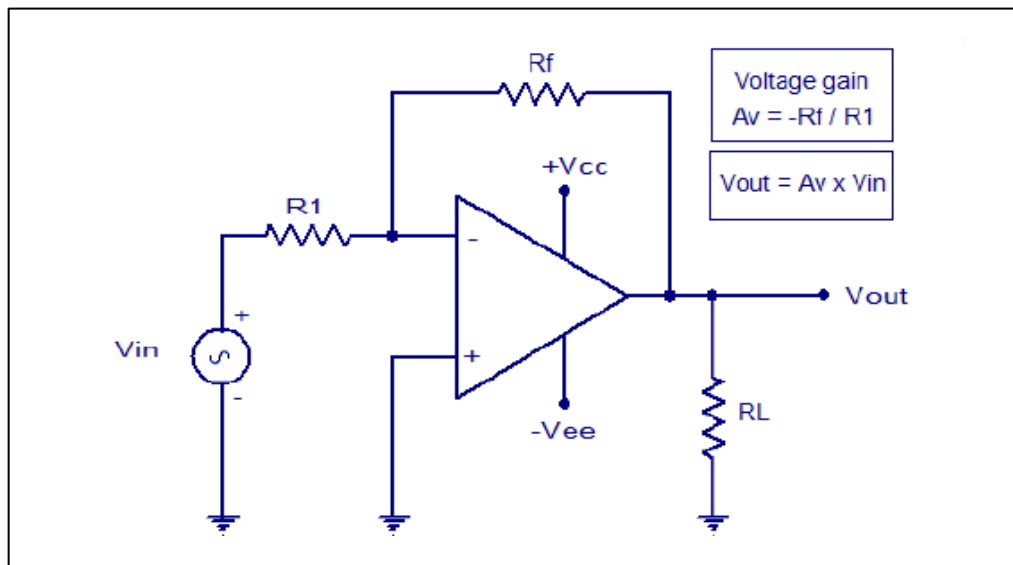


Figure 7.1.1: Inverting Amplifier

7.2 Design of Non-Inverting amplifier with OP AMP741C for gain of 11.

Plot the waveforms, observe the phase reversal, measure the gain for the circuit shown in Figure.7.2.1.

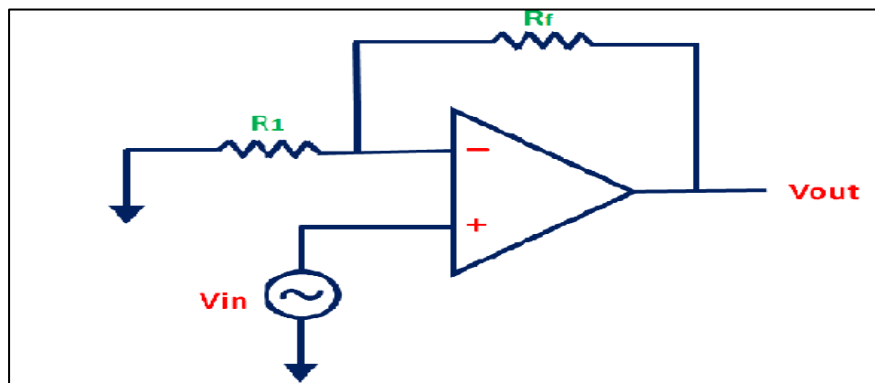


Figure 7.2.1: Inverting Amplifier

7.3. What is the output of the summing amplifier in Figure 7.3.1, with the given DC input voltages? Compare the values obtained from Simulation and Practical values.

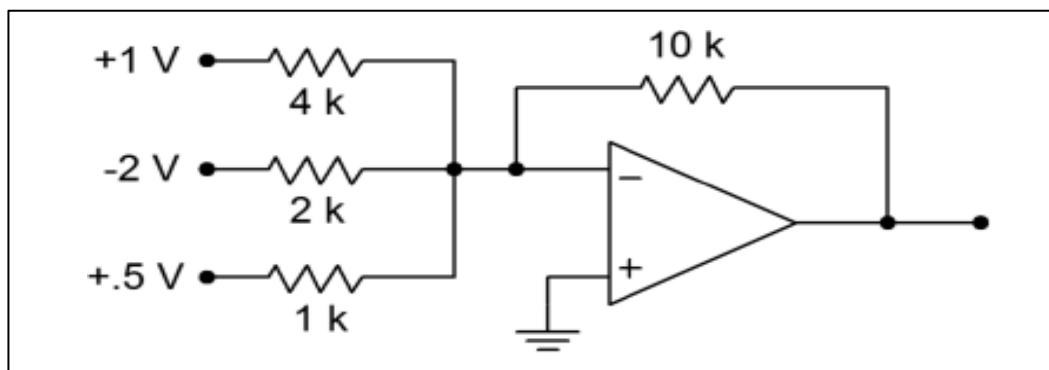


Figure.7.3.1

Hint: The easy way to approach this is to just treat the circuit as three inverting voltage amplifiers, and then add the results to get the final output.

Try

1. Design a simple difference amplifier with an input impedance of 10 k per leg, and a voltage gain of 26 Db.
2. Design Adder/ Subtractor Circuits.

Hint: If inverting and non- inverting summing amplifiers are combined using the differential amplifier topology, an adder/subtractor results. Normally, all resistors in an adder/subtractor are the same value.

8. Integrators and Differentiators

The integrator circuit is mostly used in analog computers, analog-to-digital converters and wave- shaping circuits. A common wave-shaping use is as a charge amplifier and they are usually constructed using an operational amplifier though they can use high gain discrete transistor configurations.

Differentiating amplifiers are most commonly designed to operate on triangular and rectangular signals. Differentiators also find application as wave shaping circuits, to detect high frequency components in the input signal.

8.1.1.Design of integrator circuit for periodic signal with a frequency of 5 kHz

Draw the input and output waveforms for different time constants for the basic integrator circuit shown in Figure.8.1.1.

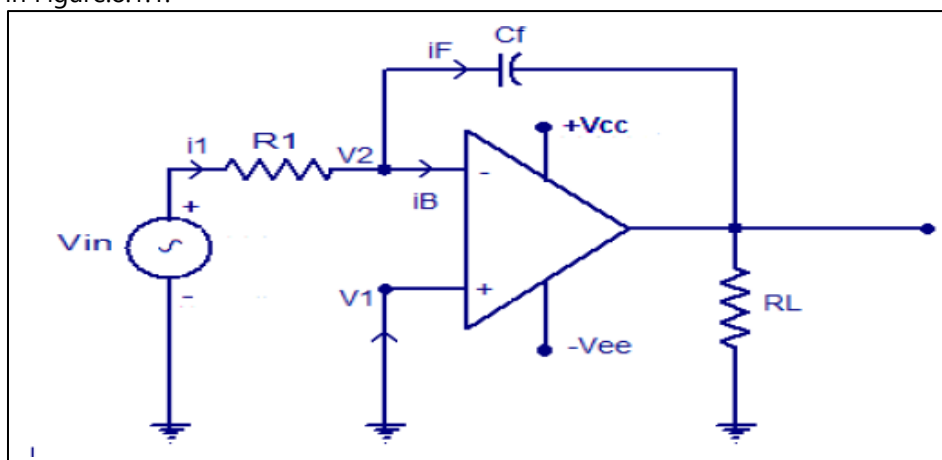


Figure 8.1.1: Basic Integrator

8.1.2 Design lossy integrator so that the peak gain is 20dB and the gain is 3dB down from its peak when $\omega = 10,000$ rad/sec. use a capacitance of $0.01\mu F$ in hardware and multisim.

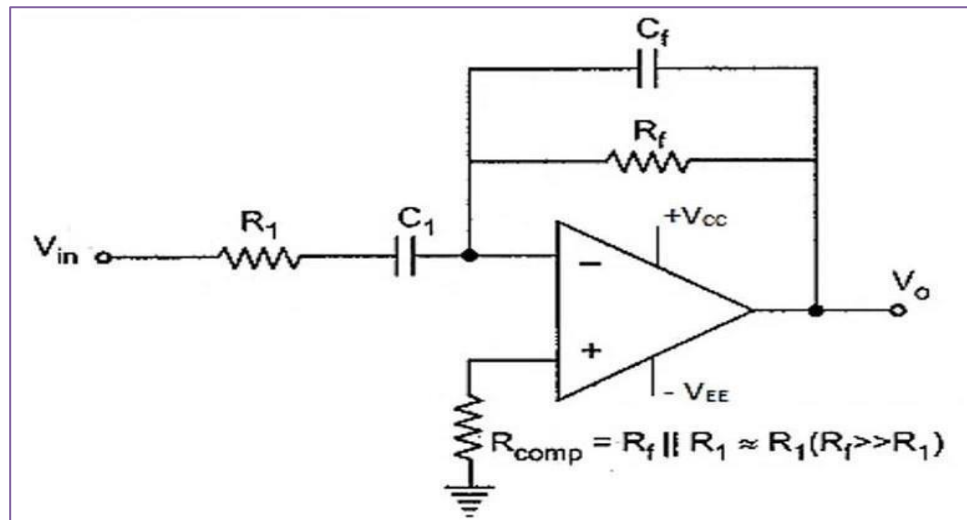


Figure 8.1.2: Practical Integrator

8.1.3. Design Summing integrator to give output as per the following equation using Multisim.

$$v_o(t) = -\frac{1}{C_f} \int_0^t \left(\frac{v_1(t)}{R_1} + \frac{v_2(t)}{R_2} + \frac{v_3(t)}{R_3} \right) dt + v_o(0)$$

8.2.1. Design of Differentiator circuit for periodic signal with a frequency of 5 kHz

Draw the input and output waveforms for different time constants for the basic Differentiator circuit shown in Figure.8.2.1.

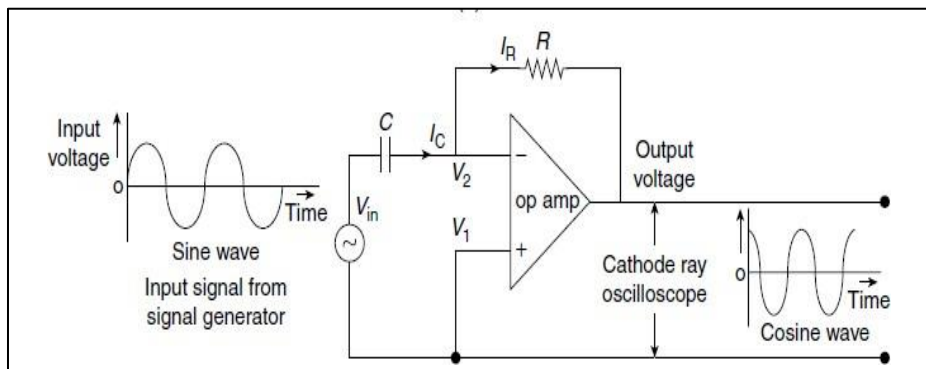


Figure 8.2.1: Basic Differentiator

8.2.2 Design a differentiator using op-amp to differentiate an input signal with $f_{max} = 200$ Hz. Also draw the output waveforms for a sine-wave and a square-wave input of 1V peak at 200 Hz. Compare the values obtained from Simulation and Practical values.

8.2.3 Construct a differentiator for a given differential input signal that varies in frequency from 100 Hz to about 5 KHz with a sine wave of 10V peak at 2k Hz is applied to this differentiator. Also draw the required waveforms.

Try

1. The circuit that integrates the input signal twice is called double integrator. The design of double integrator requires two reactive portions for obtaining double integration. Design double integrator in mutisim /pspice to produce output as per the equation as shown below.
2. Elaborate practical integrator circuit and state the advantages of practical integrator over ideal integrator. Using simulation with multisim determine R1 and Rf for a given lossy integrator for a peak gain of 30dB and the gain is 3dB down from its peak when $\omega = 5000$ rad/sec with a capacitance of 0.01micro farads.

$$v_o(t) = -\frac{4}{(RC)^2} \iint v_i(t) dt$$

3. Sketch the output waveform of a differentiator for a sinusoidal signal of 3V peak at 250Hz. Compare the values obtained from Simulation and Practical values.
4. Determine the useful range for differentiation in the circuit of Figure 8.2.3. Also determine the output voltage if the input signal is a 2 V peak sine wave at 3 kHz.

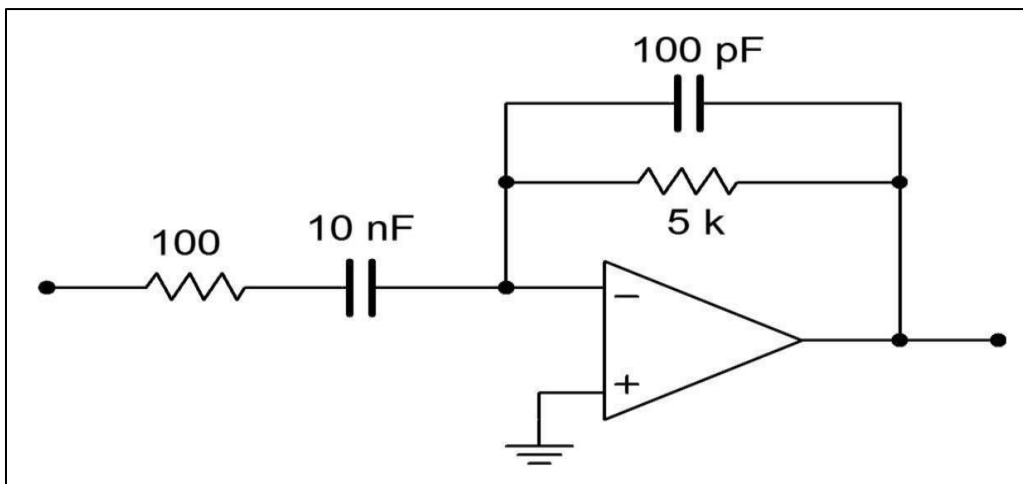


Figure 8.2.3: Differentiator

5. Given the circuit of Figure 8.2.4 carryout the simulation and analyse the output waveforms if the input is a 100 Hz, 1 V peak triangle wave.

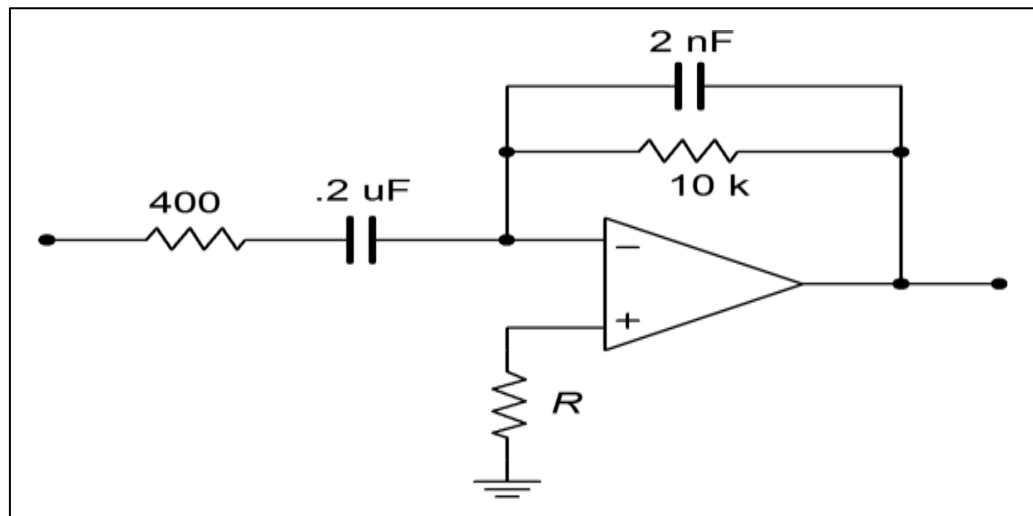


Figure 8.2.4: Differentiator

9. Exercises on Basic Logic Gates

AND, OR and NOT gates are basic gates. NAND and NOR are universal gates. Basically logic gates are electronic circuits because they are made up of number of electronic devices and components. Inputs and outputs of logic gates can occur only in two levels. These two levels are term HIGH and LOW, or TRUE and FALSE, or ON AND off, OR SIMPLY 1 AND 0. A table which lists all possible combinations of input variables and the corresponding outputs is called a „truth table“. It shows how the logic circuit's output responds to various combinations of logic levels at the inputs.

9.1.1 Basic Gates

- 9.1.1.1. Install Xilinx vivado on your machine.
- 9.1.1.2. Write a VHDL program using vivado simulator for:
- 9.1.1.3. Verifying the functionality of design under test (DUT) by writing test bench to pass the stimulus
- 9.1.1.4. Synthesize the register transfer logic (RTL) using Xilinx XST synthesis tool
- 9.1.1.5. Elaborate the design and generate bit file to dump RTL code into the zynq series and Z-board FPGA
- 9.1.1.6. Verify the functionality of the design under test (DUT) on FPGA board

9.1.2 AND-OR-Invert AND OR-AND-Invert Logic

Write VHDL code to implement the function expressed by the following logic equation

$$Y = \bar{a}\bar{0}\bar{b}\bar{0} + \bar{a}\bar{1}\bar{b}\bar{1} + \bar{a}\bar{2}\bar{b}\bar{2}$$

Use only simple signal assignment statements in your VHDL data flow model.

Hints

Use and, or and compliment operators for implementation of the logic.

Input a0, a1, a2, a3, b0, b1, b2, b3;

Output y;

```
/**    Data flow model for AOI logic */
```

```
entity AOI port (  
a0, a1, a2, a3, b0, b1, b2, b3: in std_logic; y: out  
std_logic);  
end AOI;
```

```
/**    architecture body */
```

```
architecture arch_AOI of AOI is
```

```
begin
```

```
Y = not ((a0 & b0) | (a1 & b1) | (a2 & b2) | (a3 & b3));
```

```
. . .
```

```
End arch_AOI;
```

```
//Write the test bench for providing the stimulus
```

```

entity tb_AOI port
end tb_AOI;

architecture arch_AOI of AOI is
signal a0, a1, a2, a3, b0, b1, b2, b3: std_logic := '0';
signal y: std_logic;

component AOI port (
a0, a1, a2, a3, b0, b1, b2, b3: in std_logic; y: out
std_logic);
end component;

begin
DUT: AOI port map (a0, a1, a2, a3, b0, b1, b2, b3, y);

a0 = '0'; b0 = '0';
a1 = '0'; b1 = '0';
a2 = '0'; b2 = '0';
a3 = '0'; b3 = '0';
wait for 10 ns;

a0 = '0'; b0 = '1';
a1 = '1'; b1 = '0';
a2 = '0'; b2 = '1';
a3 = '1'; b3 = '0';
wait for 10 ns;

. . .
. . .

a0 = '1'; b0 = '1';
a1 = '1'; b1 = '1';
a2 = '1'; b2 = '1';
a3 = '1'; b3 = '1';
wait for 10 ns; End

```

architecture

// After post simulation

Simulate the design using Xilinx software

Plot the wave forms and verify the functionality of the design

Synthesize the design

Elaborate the design and dump the bit file into FPGA

[Try](#)

1. Develop a model for a general or- and-invert gate, with two std_logic_vector input ports a and b and a standard-logic output port y.

9.1.2.1. Product of Sum Boolean expression

Write a program to perform product of sum which evaluates the given Boolean expressions and write the test-bench to verify the functionality with all possible combinations of the input.

Hints

```
/**
 * Consider the POS expression for  $F = \pi(0, 1, 2, 4, 8, 9, 10, 12, 13)$  */
Input a, b, c, d;
Output F;

Minimize the logic for the given F //POS logic
Implement the logic using basic gates

/**      Declare the port signals */ entity
POS port (
a, b, c, d: in std_logic; f: out std_logic);
end POS;

/**      architecture body */
architecture arch_POS of POS is

Begin
Y = F(a, b, c, d);
...

End arch_POS;

//Write the test bench for providing the stimulus entity tb_POS

port
end tb_POS;

architecture arch_POS of POS is
signal a, b, c, d: std_logic := '0';
signal f: std_logic;

component POS port (
a, b, c, d: in std_logic; f: out std_logic);
end component;

begin
DUT: POS port map (a, b, c, d, f);

a = '0'; b = '0'; c = '0'; d = '0';
wait 10 ns;

a = '0'; b = '0'; c = '0'; d = '1';
wait 10 ns;
```



```
a = '1'; b = '1'; c = '1'; d = '1';
wait 10 ns; End
```

```
architecture
```

Provide the stimulus for all 16 possible combinations starting from 0000 to 1111

Simulate the DUT with the given stimulus

Verify the output using waveforms

Synthesize the design

Elaborate the design and dump the bit file into FPGA

Try

1. Modify the POS expression including the don't care cases $F = \pi (0, 1, 2, 4, 9, 10, 12, 13) + d (3, 7, 11)$

9.1.2.2. Sum of Product Boolean expression

Write a program to perform sum of product which evaluates the given Boolean expressions and write the test-bench to verify the functionality with all possible combinations of the input.

Hints

```
/**
 * Consider the POS expression for  $F = \Sigma (0, 1, 2, 4, 8, 9, 10, 12, 13)$  */
Input a, b, c, d;
Output F;

Minimize the logic for the given F           // POS logic
Implement the logic using basic gates

/**  Declare the port signals */
entity SOP port (
a, b, c, d: in std_logic;
f: out std_logic);
end SOP;
/**  architecture body */
architecture arch_SOP of SOP is

Begin
Y = F (a, b, c, d);
. . .

End arch_SOP;

//Write the test bench for providing the stimulus

entity tb_SOP port
end tb_SOP;

architecture arch_SOP of SOP is
signal a, b, c, d: std_logic := '0';
signal f: std_logic;

component SOP port (
```

```
a, b, c, d: in std_logic; f: out
std_logic);
end component;
```

```
begin
DUT: SOP port map (a, b, c, d, f);
```

```
a = '0'; b = '0'; c = '0'; d = '0';
wait 10 ns;
```

```
a = '0'; b = '0'; c = '0'; d = '1';
wait 10 ns;
```

```
. . .
. . .
```

```
a = '1'; b = '1'; c = '1'; d = '1';
wait 10 ns; End
```

architecture

Simulate the DUT with the given stimulus

Verify the output using waveforms

Synthesize the design

Elaborate the design and dump the bit file into FPGA

Try

1. Modify the POS expression including the don't care conditions $F = \Sigma (0, 1, 2, 4, 9, 10, 12, 13) + d (3, 7, 11)$

9.1.2.3. Code Conversions

To familiarize students with code converters. The student should also become familiar with gray to binary conversion, binary to gray conversion.

Given the sequence of three-bit Gray code as (000, 001, 011, 010, 110, 111, 101, 100)

A given Gray code number can be converted into its binary equivalent by going through the following steps:

1. Begin with the most significant bit (MSB). The MSB of the binary number is the same as the MSB of the Gray code number.
2. The bit next to the MSB (the second MSB) in the binary number is obtained by adding the MSB in the binary number to the second MSB in the Gray code number and disregarding the carry, if any.
3. The third MSB in the binary number is obtained by adding the second MSB in the binary number to the third MSB in the Gray code number. Again, carry, if any, is to be

Hints

```
/**   Declare the port signals */
entity gray2binary port (
gray_code: in std_logic_vector(3 downto 0);
binary_code: out std_logic_vector(3 downto 0)); end
gray2binary;

/**   architecturebody */
architecture arch_gray2binary of gray2binary is

Begin
binary_code[3] = gray_code[3];
binary_code[2] = gray_code[3] xor gray_code[2];
. . .

End arch_gray2binary;

//Write the test bench for providing the stimulus

entity tb_gray2binary port
end tb_gray2binary;

architecture arch_gray2binary of gray2binary is
signal gray_code: std_logic_vector(3 downto 0) := ,0000';
signal binary_code: std_logic_vector(3 downto 0);

component gray2binary port (
gray_code: in std_logic_vector(3 downto 0); binary_code:
out std_logic_vector(3 downto 0));
end component;

begin
DUT: gray2binary port map (gray_code, binary_code);

Process
begin
gray_code = gray_code + 1; Wait
for 10 ns;

End process; End

architecture

// After post simulation

Simulate the design using Xilinx software

Plot the wave forms and verify the functionality of the

design Synthesize the design
```

Try

1. Design and implement the binary to gray code conversion
2. Design and implement the binary to excess 3-code conversion
3. Design and implement the excess 3-code to binary conversion

9.1.3 Introduction

NAND gate is universal gate. It can perform all the basic logic function. NAND means NOT AND that is, AND output is NOTed. so NAND gate is combination of an AND gate and a NOT gate. The output is logic 0 level, only when each of its inputs assumes a logic 1 level. For any other combination of inputs, the output is logic 1 level. NAND gate is equivalent to a bubbled OR gate.

NOR gate is universal gate. It can perform all the basic logic function. NOR means NOT OR that is, OR output is NOTed. so NOR gate is combination of an OR gate and a NOT gate. The output is logic 1 level, only when each of its inputs assumes a logic 0 level. For any other combination of inputs, the output is logic 0 level. NOR gate is equivalent to a bubbled AND gate.

9.1.3.1. Basic gates realization using NAND gate

Realize the inverter gate logic using NAND gate.

NAND gate is actually a combination of two logic gates i.e. AND gate followed by NOT gate. So its output is complement of the output of an AND gate. This gate can have minimum two inputs. By using only NAND gates, we can realize all logic functions: AND, OR, NOT, Ex-OR, Ex-NOR, NOR. So this gate is also called as universal gate.

Hint

```
/**   Declare the port signals */
entity inv_nand port (
i: in std_logic; y: out
std_logic);
end inv_nand;

/**   architecture body */
architecture arch_inv_nand of inv_nand is

component nand_gate port (
a, b: in std_logic; y: out
std_logic);
end component;

begin
DUT: nand_gate port map (i, i, y);
. . .

End arch_inv_gate;

//Write the test bench for providing the stimulus

entity tb_inv_nand port
end tb_inv_nand;

architecture arch_inv_nand of inv_nand is
signal i: std_logic := '0';
signal y: std_logic;

component inv_nand port ( i:
in std_logic; y: out
std_logic);
end component;

begin
```

```

DUT: inv_nand port map (i, y);

i ='0' after 10 ns;
i ='1' after 10 ns;

. . .
. . .

end architecture

```

Try

Realize the AND gate logic using NAND gate

Realize the OR gate logic using NAND gate

9.1.3.2 Gate realization using NOR

Realize the inverter gate logic using NOR gate. **Hint**

```

/**    Gate level model for */
entity inv_nor port (
i: in std_logic; y: out
std_logic);
end inv_nor;

/**    architecture body */
architecture arch_inv_nor of inv_nand is

component nor_gate port (
a, b: in std_logic; y: out
std_logic);
end component;

begin
DUT: nor_gate port map (i, i, y);
. . .

End arch_inv_nor;

//Write the test bench for providing the stimulus

entity tb_inv_nor port
end tb_inv_nor;

architecture arch_inv_nor of inv_nor is
signal i: std_logic := '0';
signal y: std_logic;

component inv_nor port ( i:
in std_logic; y: out
std_logic);
end component;

begin
DUT: inv_nor port map (i, y);

i ='0' after 10 ns;

```

```
i ='1' after 10 ns;
```

```
. . .  
. . .
```

```
End architecture
```

Try

1. Realize the AND gate logic using NOR gate
2. Realize the OR gate logic using NOR gate

9.1.3.3 XOR gate realization using minimum number of NAND gates

Realize XOR gate using minimum number of NAND gates.

Hint

```
/**   Declare the port signals */  
entity xor_nand port (  
a, b: in std_logic; y: out  
std_logic);  
end xor_nand;  
  
/**   architecture body */  
architecture arch_xor_nand of xor_nand is  
  
component nand_gate port (  
a, b: in std_logic; y: out  
std_logic);  
end component;  
  
begin  
DUT: nand_gate port map (a, b, y);  
. . .  
  
End arch_xor_nand;  
  
//Write the test bench for providing the stimulus  
  
entity tb_xor_nand port  
end tb_xor_nand;  
  
architecture arch_xor_nand of xor_nand is  
signal a, b: std_logic := '0';  
signal y: std_logic;  
  
component xor_nand port ( a,  
b: in std_logic; y: out  
std_logic);  
end component;  
  
begin  
DUT: xor_nand port map (a, b, y);  
  
a ='0' after 10 ns;  
b ='0' after 10 ns;
```

```
. . .  
. . .
```

End architecture

Try

1. Realize XNOR gate using minimum number of NAND gates

9.1.3.4 Three input NAND gate using min no of 2 input NAND Gate

Implement 3 input NAND gate realization using minimum number of NAND gates.

And B to the first NAND gate

Output of first Nand gate is given to the two inputs of the second NAND gate (this basically realizes the inverter functionality)

Output of second NAND gate is given to the input of the third NAND gate, whose other input is C ((A NAND B) NAND (A NAND B)) NAND C Thus, can be implemented using '3' 2-input NAND gates.

Hints:

Assume three inputs of the NAND gate are A, B and C and connect these inputs as

```
/**    Declare the port signals */  
entity nand3 port (  
  a, b, c: in std_logic; y: out  
  std_logic);  
end nand3;  
  
/**    architecture body */  
architecture arch_nand3 of nand3 is  
  
  component nand2_gate port (  
    a, b: in std_logic;  
    y: out std_logic); end  
  component;  
  
  begin  
    DUT: nand2_gate port map (a, b, y);  
    . . .  
  
  End arch_nand3;  
  
  //Write the test bench for providing the stimulus  
  
  entity tb_nand3 port  
  end tb_nand3;  
  
  architecture arch_tb_nand3 of tb_nand3 is  
    signal a, b: std_logic := '0';  
    signal y: std_logic;  
  
    component nand2 port (  
      a, b: in std_logic;
```

```

y: out std_logic); end
component;

begin
DUT: nand2 port map (a, b, y);

a = '0' after 10 ns;
b = '0' after 10 ns;

. . .
. . .

a = '1' after 10 ns;
b = '1' after 10 ns; End

architecture

```

Try:

Implement XNOR gate realization using minimum number of NAND gates

9.1.3.5 User defined logic gate (Muller-C element cell)

Develop a behavioral model for a two-input Muller-C element cell, with two input ports and one output, all of type bit. The inputs and outputs are initially „0“. When both inputs are „1“, the output changes to „1“. It stays „1“ until both inputs are „0“, at which time it changes back to „0“. Your model should have a propagation delay for rising output transitions of 3.5 ns, and for falling output transitions of 2.5 ns.

Hints:

Assume three inputs of the NAND gate are A, B and C and connect these inputs as,

```

Take inputs A and B
Extract the truth table and Boolean expression as per the specifications
Implement the gate using VHDL model
Write the logic for selecting the stimulus for verifying the logic

/**    Declare the port signals */
entity mc_cell port (
a, b, c: in std_logic; y: out
std_logic);
end mc_cell;

/**    architecture body */
architecture arch_mc_cell of mc_cell is

component mc_cell port (
a, b: in std_logic; y: out
std_logic);
end component;

begin

```



```

DUT: nand2_gate port map (a, b, y);
. . .

End arch_mc_cell;

//Write the test bench for providing the stimulus

entity tb_mc_cell port
end tb_mc_cell;

architecture arch_tb_mc_cell of tb_mc_cell is
signal a, b: std_logic := '0';
signal y: std_logic;

component mc_cell port ( a,
b: in std_logic; y: out
std_logic);
end component;

begin
DUT: mc_cell port map (a, b, y);

a ='0' after 10 ns;
b ='0' after 10 ns;

. . .
. . .

a = '1' after 10 ns;
b = '1' after 10 ns; End

architecture

Plot the wave form for all possible select lines
Synthesize the design
Elaborate the design and dump the bit file into FPGA

```

Try

Develop a behavioral model for a two-input Muller-C element cell, with two input ports and one output, all of type bit. The inputs and outputs are initially „1“. When both inputs are „0“, the output changes to „0“. It stays „0“ until both inputs are „1“, at which time it changes back to „1“.

10. Exercises on Adder / Subtractor

In digital electronics, adders and subtractors both are the combinational logic circuits (a combinational logic circuit is one whose output depends only on the present inputs, but not on the past outputs) that can add or subtract numbers, more specifically binary numbers. Adders and subtractors are the crucial parts of arithmetic logic circuits in processing devices like microprocessors or microcontrollers.

A combinational logic circuit which is designed to add two binary digits is called as a half adder. The half adder provides the output along with a carry value (if any). The half adder circuit is designed by connecting an EX-OR gate and one AND gate. It has two input terminals and two output terminals for sum and carry.

10.1.Implementation of half adder

Design a gate level circuit for half adder and verify for the following truth table and write a test bench for verifying the functionality of the ha1f adder.

Consider	the	inputs are	A, B and outputs are Sum, Cout
A	B	Sum	Cout
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Hints

The code pattern for implementing half adder:

```
// Declare the port signals
Input a, b;
Output sum, cout

//Write VHDL model for half adder in behavioral flow model
entity ha_df port (
a, b: in std_logic;
sum, cout: out std_logic); end ha_df;

/**    architecture body */
architecture arch_ha_df of ha_df is

begin
process(a, b) begin

case {a, b} is
when ,00' => sum <= '0'; cout <= '0';
when ,01' => sum <= '1'; cout <= '0';
when ,10' => sum <= '1'; cout <= '0';
. . .
. . .
. . .
End case End
process;

End arch_ha_df;

//Write the test bench for providing the stimulus

entity tb_ha_df port
end tb_ha_df;

architecture arch_tb_ha_df of tb_ha_df is
signal a, b: std_logic := '0';
signal sum, cout: std_logic;
```

```

component ha_df port (
a, b: in std_logic;
sum, cout: out std_logic); end
component;

begin
DUT: ha_df port map (a, b, sum, cout);

process
begin
a = ~a;
wait for 10ns;
b = ~b;
wait for 15ns; end

process

End architecture

Plot the wave form for all possible select lines
Synthesize the design
Elaborate the design and dump the bit file into FPGA

```

10.1.1.1 Implementation of full adder

Design a gate level circuit for full adder and verify for the following truth table and write a test bench for verifying the functionality of the full adder.

Consider the inputs are A, B, C				and outputs are Sum, Cout	
A	B	C	Sum	Cout	
0	0	0	0	0	
0	0	1	1	0	
0	1	0	1	0	
0	1	1	0	1	
1	0	0	1	0	
1	0	1	0	1	
1	1	0	0	1	
1	1	1	1	1	

Hints

The pseudo code for printing full adder:

```

end fa_gl;

/** architecture body */ architecture arch_fa_gl
of ga_gl is
component xor_gate port ( a, b: in
std_logic; y: out std_logic);
end component;

component declaration for or_gate, and_gate begin

u0: xor_gate port map(a, b, x1);
u1: xor_gate port map(x1, cin, sum);
...
...

End arch_fa_gl;

//Write the test bench for providing the stimulus entity tb_fa_gl

port
end tb_fa_gl;

architecture arch_tb_fa_gl of tb_fa_gl is signal a, b, cin:
std_logic := '0'; signal sum, cout: std_logic;

component fa_gl port (
a, b, cin: in std_logic; sum, cout: out std_logic);
end component;

begin
DUT: fa_gl port map (a, b, cin, sum, cout);

process begin
a = ~a;
wait for 5ns; b = ~b;
wait for 10ns; cin = ~cin;
wait for 15ns;

end process End

architecture

```

Plot the wave form for all possible select lines Synthesize the design
Elaborate the design and dump the bit file into FPGA

10.1.1.2 Realization of full adder using half adders

Design a gate level circuit for full adder using half adder and verify for the functionality and write a test bench for verifying the functionality of the full adder.

Hints

The pseudo code for printing full adder:

```
// Declare the port
signals Input a, b, c;
Output sum, cout

Work lib should consists of half adder, or_gate

modules entity fa_ha port (
a, b, cin: in std_logic; sum, cout:
out std_logic);
end fa_ha;

/**    architecture body */
architecture arch_fa_ha of ga_ha is
component ha_df port (
a, b: in std_logic; y: out
std_logic);
end component;

component declaration for or_gate, and_gate
begin

u0: xor_gate port map(a, b, x1);
u1: xor_gate port map(x1, cin, sum);
. . .
. . .

End arch_fa_gl;

//Write the test bench for providing the stimulus

entity tb_fa_gl port
end tb_fa_gl;

architecture arch_tb_fa_gl of tb_fa_gl is
signal a, b, cin: std_logic := '0';
signal sum, cout: std_logic;

component fa_gl port (
a, b, cin: in std_logic;
sum, cout: out std_logic); end
component;

begin
DUT: fa_gl port map (a, b, cin, sum, cout);
```

```

begin
a = ~a;
wait for 5ns; b =
~b;
wait for 10ns; cin =
~cin; wait for 15ns;

end process End

architecture

Plot the wave form for all possible select lines
Synthesize the design
Elaborate the design and dump the bit file into FPGA

```

10.1.1.3 Design and implement 4-bit ripple carry adder

Design a gate level circuit for ripple carry adder using full adder and verify for the functionality and write a test bench for verifying the functionality of the ripple carry adder.

Hints

The pseudo code for printing full adder:

```

// Declare the port signals
Input a, b;      //vector of 4-bit size
Input cin;
Output sum;      //vector of 4-bit size
Output cout;

// Declare xor gate, and gate
Component declaration for full adder

// instance xor gate, and gate
Port map for full adder

```

10.1.1.4 Implementation of half subtractor

Design a gate level circuit for half subtractor and verify for the following truth table and write a test bench for verifying the functionality of the half subtractor.

Consider the inputs are A, B and outputs are Diff, Bout

A	B	Diff	Bout
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Hints

The code pattern for printing half subtractor:

```
// Declare the port signals
Input a, b;
Output diff, bout

// Declare xor gate, and gate
Component declaration for xor gate, and gate

// instance xor gate, and gate
Port map for xor gate
Port map for and gate
```

10.1.1.5Implementation of full subtractor

Design a gate level circuit for full subtractor and verify for the following truth table and write a test bench for verifying the functionality of the full subtractor.

Consider	the	inputs are	A, B,	Bin and outputs	are	Diff,	Bout
A	B	Bin	Sum	Bout			
0	0	0	0	0			
0	0	1	1	1			
0	1	0	1	1			
0	1	1	0	1			
1	0	0	1	0			
1	0	1	0	0			
1	1	0	0	0			
1	1	1	1	1			

Hints

The pseudo code for printing full subtractor:

```
// Declare the port signals
Input a, b, bin;
Output diff, bout

// Declare xor gate, and gate
Component declaration for xor gate, and gate

// instance xor gate, and gate
Port map for xor gate
Port map for and gate
```

10.1.1.6 Realization of full subtractor using half subtractor

Design a gate level circuit for full subtractor using half subtractor and verify for the functionality and write a test bench for verifying the functionality of the full subtractor.

Hints

The pseudo code for printing full subtractor:

```
// Declare the port signals
Input a, b, bin;
Output diff, bout
// Declare xor gate, and gate
Component declaration for half subtractor, and gate, or gate
// instance xor gate, and gate
Port map for half subtractor
Port map for and gate
Port map for or gate
```

Try

Design a gate level circuit for ripple carry adder using full adder and verify for the functionality and write a test bench for verifying the functionality of the half adder.

Hints

The pseudo code for printing full adder:

```
// Declare the port signals
Input a, b;          //vector of 4-bit size
Input cin;
Output sum;          //vector of 4-bit size
Output cout;

// Declare xor gate, and gate
Component declaration for full adder

// instance xor gate, and gate
Port map for full adder
```

11. Exercises on Multiplexer, Decoders

In digital systems, many times it is necessary to select a single data line from several data-input lines and the data from the selected data input line should be available on the output line. The digital circuit which does this task is a multiplexer.

11.1. Implementation of 2x1, 4x1 multiplexers

Develop a behavioral model for a two-input multiplexer, with ports of type bit and a propagation delay from data or select input to data output of 5 ns. You should declare a constant for the propagation delay, rather than writing it as a literal in signal assignments in the model.

The inputs to the MUX are data inputs I1, I0 and a one control input SEL(s) The single output is Y.

Hints

```
/**      Implementation of 2x1 multiplexer      **/

Declare the inputs I0, I1 and S
Declare the output Y.

//Write the logic for selecting the data depends      on the select line and
pass to the output

entity mux_2x1 port (
i0, i1, s: in std_logic; y: out
std_logic);
end mux_2x1;

/**      architecture body */
architecture arch_mux_2x1 of mux_2x1 is

component nand port (
a, b: in std_logic; y: out
std_logic);
end component;

component inv port (
i: in std_logic; y: out
std_logic);
end component;

begin
DUT1: inv port map (s, sb);
DUT2: nand2_gate port map (i0, sb, s1); . . .

. . .
. . .

End arch_mux2x1;

//Write the test bench for providing the stimulus

entity tb_mux2x1 port
end tb_mux2x1;

architecture arch_tb_mux2x1 of tb_mc_mux2x1 is
signal i0, i1, s: std_logic := '0'; signal
y: std_logic;

component mux2x1 port (
i0, i1, s: in std_logic; y: out
std_logic);
end component;
```

```

begin
DUT: mux2x1 port map (i0, i1, s, y);

s ='0' after 10 ns, '1' after 10 ns;
process
begin
i0 = ~i0; wait for
10ns; i1 = ~i1;
wait for 15ns;

end process End

architecture

Plot the wave form for all possible select lines
Synthesize the design
Elaborate the design and dump the bit file into FPGA

Plot the wave form for all possible select lines
Synthesize the design
Elaborate the design and dump the bit file into FPGA

```

Try

Develop a behavioral model for a four-input multiplexer, with ports of type bit and a propagation delay from data or select input to data output of 4.5 ns. You should declare a constant for the propagation delay, rather than writing it as a literal in signal assignments in the model.

11.2 Implementation of 2x1, 4x1 demultiplexers

Build a behavioral model for a two-input demultiplexers, with ports of type bit and a propagation delay from data or select input to data output of 5 ns.

Hints

```

/**
Implementation of 2x1 demultiplexer.
**/

Declare the inputs I and S
Declare the output Y0, Y1.

//Write the logic for selecting the data depends on the select line and
pass to the output

entity dmux_1x2 port (
i, s: in std_logic; y0, y1: out
std_logic);
end dmux_1x2;

```

```

/**      architecture body */
architecture arch_dmux_1x2 of dmux_1x2 is

begin
process(l,s) begin
case S is
when '0': y0 = l; y1 = 'z';
...
...

End arch_dmux1x2;

//Write the test bench for providing the stimulus entity

tb_dmux1x2 port
end tb_dmux1x2;

architecture arch_tb_dmux1x2 of tb_dmux1x2 is
signal i, s: std_logic := '0';
signal y0, y1: std_logic;

component dmux1x2 port ( i,s: in
std_logic;
y0,y1: out std_logic); end component;

begin
DUT: dmux1x2 port map (i, s, y0, y1);

s ='0' after 10 ns, '1' after 10 ns; process
begin
i = ~i;
wait for 10ns; end process

End architecture

```

Plot the wave form for all possible select lines Synthesize the design

Elaborate the design and dump the bit file into FPGA

Plot the wave form for all possible select lines Synthesize the design

Elaborate the design and dump the bit file into FPGA

Try

- 1 Modify the program to function as 1x4 demultiplexers
- 2 Modify the program to function as 1x8 demultiplexers

11.3 Realization of higher order multiplexers using lower order multiplexers

Realize the higher order multiplexers using lower order multiplexers. Write the VHDL model for the realized circuits. Simulate the test benches for the corresponding and verify the design under test by plotting the waveforms. Figure 11.3 shows realization of 4x1 mux using 2x1 mux.

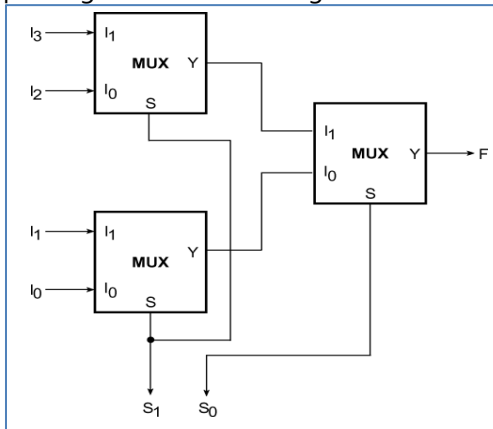


Figure 11.3: Realization of 4x1 mux using 2x1 mux

```
/**
Realize the lower order multiplexers for design of higher order multiplexers.
**/
```

In work library simulate the lower order multiplexer

For the realized higher order multiplexer, instance the component in the declaration part of the architecture

By using positional or name mapping instance the component with the signals

Hint

```
//Write the logic for selecting the data depends on the select line and
pass to the output
```

```
entity mux_4x1 port (
i0, i1, i2, i3: in std_logic;
s: in std_logic_vector(1 downto 0); y: out
std_logic);
end mux_4x1;
```

```
/** architecture body */
architecture arch_mux_4x1 of mux_4x1 is
```

```
component mux2x1 port (
i0, i1, s: in std_logic; y: out
std_logic);
end component;
// declare intermediate signals
```

```
begin
DUT1: mux2x1 port map (i0, i1, s(0), s1);
```

```

DUT2: mux2x1 port map (i2, i3, s(1), s2);
. . .
. . .
. . .

End arch_mux4x1;

//Write the test bench for providing the stimulus

entity tb_mux4x1 port
end tb_mux4x1;

architecture arch_tb_mux4x1 of tb_mux4x1 is
signal i0, i1, i2, i3: std_logic := '0';
signal s: std_logic_vector(1 downto 0) :=
,00'; signal y: std_logic;

component mux4x1 port (
i0, i1, i2, i3: in std_logic;
s: in std_logic_vector(1 downto 0); y:
out std_logic);
end component;

begin
DUT: mux4x1 port map (i0, i1, i2, i3, s, y);

s ='00' after 10 ns, '01' after 10 ns . . .;
process
begin
i0 = ~i0; wait for
10ns; i1 = ~i1;
wait for 12ns; i2
= ~i2; wait for
14ns; i3 = ~i3;
wait for 16ns;
end process End

architecture

Plot the wave form for all possible select lines
Synthesize the design
Elaborate the design and dump the bit file into FPGA

```

11.4 Realization of basic gates using 2x1 multiplexer

Realize all the basic gates like AND and inverter using 2x1 multiplexer.

```

/**      Realize the multiplexers for function as basic logic
gates */ In work library simulate the 2x1 multiplexer

```

For the realized basic gates using 2x1 multiplexer, instance the component in the declaration part of the architecture

Hints

//Write the logic for selecting the data depends on the select line and pass to the output

```
entity mux_and port (  
a,b: in std_logic; y: out std_logic;  
end mux_2x1;  
  
/**      architecture body */ architecture  
arch_mux_and of mux_and is
```

```
component mux2x1 port (  
i0,i1,s: in std_logic; y: out std_logic);  
end component;  
// declare intermediate signals
```

```
begin  
DUT1: mux2x1 port map ('0', b, a, y);  
...  
...  
...  
End arch_mux_and;
```

//Write the test bench for providing the stimulus entity

```
tb_mux_and port  
end tb_mux_and;
```

```
architecture arch_tb_mux_and of tb_mux_and is  
signal a, b: std_logic := '0';  
signal y: std_logic;
```

```
component mux_and port ( a, b: in  
std_logic; y: out std_logic);  
end component; begin  
DUT: mux_and port map (a, b, y);
```

```
process begin  
a = ~a;  
wait for 10ns; b = ~b;  
wait for 12ns; end process
```

End architecture

Plot the wave form for all possible select lines Synthesize the design

Elaborate the design and dump the bit file into FPGA

Try

11.4.1 Realize all the OR gate using 2x1 multiplexer

11.4.2 Realize all the basic gates like XOR and XNOR using 2x1 multiplexer

11.4.3 Realize all the inverter using 2x1 multiplexer

11.5 Introduction

A decoder is a multiple input multiple output logic circuit which converts coded input into coded output where input and output codes are different. The input code generally has fewer bits than the output code. Each input code word produces a different output code word i.e there is one to one mapping can be expressed in truth table. In the block diagram of decoder circuit the encoded 2^n information is present as n input producing 2^n through output $2^n - 1$.

11.5.1.1 Implementation of 2 to 4 decoder

Write the VHDL code for the circuit contains an input bundle of two input signals and an output bundle of four decoded signals. The input bundle, i_0 , i_1 represents decoder inputs. The output bus, $Y0$, $Y1$, $Y2$ and $Y3$, are used to indicate the decoded output for the two inputs. The relationship between the input and output is shown in the table below. Use a selected signal assignment statement in the solution.

```

. . .
. . .

End arch_dec2to4;

//Write the test bench for providing the stimulus

entity tb_dec2to4 port
end tb_dec2to4;

architecture arch_tb_dec2to4 of tb_dec2to4 is
signal i0, i1: std_logic := '0';
signal y0, y1, y2, y3: std_logic;

component dec2x4 port (
i0, i1: in std_logic;
y0, y1, y2, y3: out std_logic); end
component;

begin
DUT: dec2to4 port map (i0,i1, y0, y1, y2, y3);

process
begin
i0 = ~i0; wait for
10ns; i1 = ~i1;
wait for 15ns;

end process End

architecture

Plot the wave form for all possible select lines
Synthesize the design
Elaborate the design and dump the bit file into FPGA

```

Try

- 1 Write a program to implement gate level model for 2 to 4 decoder. Plot the waveforms
- 2 Write a program to implement behavioral model for 2 to 4 decoder. Plot the waveforms

11.6 Implementation of 3 to 8 decoder

Build behavioral model to function as 3 to 8 decoder.

Hints

```

/** Behavioral model implementation of 3 to 8 decoder    */
Declare the inputs I0, I1, I2.
Declare the output Y0, Y1, Y2, Y3, Y4, Y5, Y6 and Y7.

//Write VHDL model for 2 to 4 decoder gate level model
entity dec3to8 port (
i0, i1, i2: in std_logic;

```



```
y0, y1, y2, y4, y5, y6, y7: out std_logic); end dec3to8;
```

```
/**      architecture body */ architecture  
arch_dec3to8 of dec3to8 is
```

```
//declare intermediate signals begin  
process(i0, i1, i2)  
begin  
{ y0, y1, y2, y4, y5, y6, y7} = ,00000000';
```

```
case {i2, i1, i0} is  
when ,000' => Y0 <= '1';  
when ,001' => Y0 <= '1';  
when ,010' => Y0 <= '1';
```

```
...  
...  
...
```

```
End case End process;
```

```
End arch_dec3to8;
```

```
//Write the test bench for providing the stimulus entity
```

```
tb_dec3to8 port  
end tb_dec3to8;
```

```
architecture arch_tb_dec3to8 of tb_dec3to8 is  
signal i0, i1, i2: std_logic := '0';  
signal y0, y1, y2, y3, y4, y5, y6, y7: std_logic;
```

```
component dec3x8 port (  
i0, i1, i2: in std_logic;  
y0, y1, y2, y3, y4, y5, y6, y7: out std_logic); end component;
```

```
begin  
DUT: dec3to8 port map (i0, i1, i2, y0, y1, y2, y3, y4, y5, y6, y7);
```

```
process begin  
i0 = ~i0; wait for 10ns; i1 =  
~i1; wait for 15ns;
```

```
end process End
```

```
architecture
```

Plot the wave form for all possible select lines Synthesize the design

Elaborate the design and dump the bit file into FPGA

Try

- 1 Construct a 4-to-16 line decoder with two 3-to-8 line decoders having active LOW ENABLE inputs.
- 2 Implement the three-variable Boolean function $F = a\bar{c} + ab\bar{c} + ab\bar{c}$ using (i) an 8-to-1 multiplexer and (ii) a 4-to-1 multiplexer.

11.7 Realization of higher order multiplexers using lower order multiplexers

Construct a 4-to-16 line decoder with two 3-to-8 line decoders having active LOW ENABLE inputs.

Hints

```
/**  
Realize the lower order multiplexers for design of higher order  
multiplexers  
**/
```

In work library simulate the lower order multiplexer

For the realized higher order multiplexer, instance the component in the declaration part of the architecture

By using positional or name mapping instance the component with the signals

Design a test bench

Write the logic for generating stimulus for the input

signals Plot the wave form for all possible select lines

Synthesize the design

Try

Construct a 3-to-8 line decoder with two 2-to-4 line decoders having active LOW ENABLE inputs.

11.7.1.1 Realization of basic gates using 2x1 multiplexer

Realize all the basic gates like AND, OR, XOR, XNOR and inverter using 2x1 multiplexer

```
/** Realize the multiplexers for function as basic logic gates **/
```

In work library simulate the 2x1 multiplexer

For the realized basic gates using 2x1 multiplexer, instance the component in the declaration part of the architecture

By using positional or name mapping instance the component with the signals

Design a test bench

Write the logic for generating stimulus for the input signals

Plot the wave form for all possible select lines

Synthesize the design

Elaborate the design and dump the bit file into FPGA

Try

Realization of BCD-to-seven-segment decoder for a light emitting diode (LED) display

Develop a functional model of a BCD-to-seven-segment decoder for a light emitting diode (LED) display. The decoder has a 4-bit input that encodes a numeric digit between 0 and 9. There are seven outputs indexed from „a“ to „g“, corresponding to the seven segments of the LED display as shown in the margin. An output bit being „1“ causes the corresponding segment to illuminate. For each input digit, the decoder activates the appropriate combination of segment outputs to form the displayed representation of the digit.

Hint:

For example, for the input "0010", which encodes the digit 2, the output is "1101101". Your model should use a selected signal assignment statement to describe the decoder function in truth-table form.

11.7.2 Introduction

An encoder is a digital circuit that performs inverse operation of a decoder. An encoder has 2^n input lines and n output lines. In encoder the output lines generate the binary code corresponding to the input value. In octal to binary encoder it has eight inputs, one for each octal digit and three output that generate the corresponding binary code. In encoder it is assumed that only one input has a value of one at any given time otherwise the circuit is meaningless. It has an ambiguity that when all inputs are zero the outputs are zero. The zero outputs can also be generated when $D_0 = 1$.

11.7.2.1 Implementation of 4 to 2 encoder

An encoder is a digital circuit that converts a set of binary inputs into a unique binary code. The binary code represents the position of the input and is used to identify the specific input that is active. Encoders are commonly used in digital systems to convert a parallel set of inputs into a serial code.

The 4 to 2 Encoder consists of four inputs Y_3, Y_2, Y_1 & Y_0 , and two outputs A_1 & A_0 . At any time, only one of these 4 inputs can be „1“ in order to get the respective binary code at the output. The figure below shows the logic symbol of the 4 to 2 encoder.

/** Implementation of 4 to 2 decoder **/

Declare the inputs I0, I1, I2, I3. Declare the output Y0 and Y1

I3	I2	I1	I0	Y1	Y0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

Hints

//Write VHDL model for 2 to 4 decoder gate level model

```
entity enc4to2 port (  
    x : in STD_LOGIC_VECTOR(3 downto 0);  
    y : out STD_LOGIC_VECTOR(1 downto 0)); end  
enc4to2;
```

architecture arch_enc4to2 of enc4to2 is begin

```
process(i) begin  
    if (i="1000") then y <= "00"; elsif (i="0100") then y <= "01";  
    elsif (i="0010") then y <= "10"; elsif (i="0001") then y <= "11";  
    else y <= "ZZ";  
end if; end process;
```

End arch_enc4to2;

//Write the test bench for providing the stimulus entity

```
tb_enc4to2 port  
end tb_enc4to2;
```

```
architecture arch_tb_enc4to2 of tb_enc4to2 is  
    signal i : in STD_LOGIC_VECTOR(3 downto 0) := ,0000';  
    signal y : out STD_LOGIC_VECTOR(1 downto 0);
```

```
component enc4to2 port (  
    x : in STD_LOGIC_VECTOR(3 downto 0);  
    y : out STD_LOGIC_VECTOR(1 downto 0)); end  
component;
```

```
begin  
    DUT: enc4to2 port map (i, y);
```

```
process begin  
    i = i + '1';  
    wait for 10ns; end process
```

End architecture

Plot the wave form for all possible select lines Synthesize the design

Elaborate the design and dump the bit file into FPGA

Try

1. Modify VHDL behavioral model with gate level model for 4 to 2 encoder. Plot the waveforms.
2. Decimal to BCD Encoder.

The decimal-to-binary encoder usually consists of 10 input lines and 4 output lines. Each input line corresponds to each decimal digit and 4 outputs correspond to the BCD code. This encoder accepts the decoded decimal data as an input and encodes it to the BCD output which is available on the output lines.

3. Octal to Binary Encoder (8 to 3 Encoder)

The 8 to 3 Encoder or octal to Binary encoder consists of 8 inputs: Y7 to Y0 and 3 outputs: A2, A1 & A0. Each input line corresponds to each octal digit and three outputs generate corresponding binary code.

11.7.2.2 Implementation of 8 to 3 priority encoder

A 8 to 3 priority encoder has eight inputs Y7, Y6, Y5, Y4, Y3, Y2, Y1 & Y0 and two outputs A2, A1 and A0. Here, the input, Y7 has the highest priority, whereas the input, Y0 has the lowest priority. In this case, even if more than one input is „1“ at the same time, the output will be the binary code corresponding to the input, which is having higher priority.

We considered one more output, V in order to know, whether the code available at outputs is valid or not.

- If at least one input of the encoder is „1“, then the code available at outputs is a valid one. In this case, the output, V will be equal to 1.
- If all the inputs of encoder are „0“, then the code available at outputs is not a valid one. In this case, the output, V will be equal to 0.

The Truth table of 4 to 2 priority encoder is shown below.

/** Implementation of 4 to 2 decoder **/					
Declare the inputs I0, I1, I2, Declare the output Y0 and Y1					
I3	I2	I1	I0	Y1	Y0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

Hints

```

//Write VHDL model for 2 to 4 decoder gate level model
entity penc8to3 port (
  x :in STD_LOGIC_VECTOR(7 downto 0);
  y :out STD_LOGIC_VECTOR(2 downto 0)); end
enc4to2;

architecture arch_penc8to3 of penc8to3 is begin

y <= "111" when i(7)='1' else "110" when
i(6)='1' else "101" when i(5)='1' else "100"
when i(4)='1' else
"011" when i(3)='1' else "010" when
i(2)='1' else

```

```

"001" when i(1)='1' else "000" ;

End arch_penc8to3;

//Write the test bench for providing the stimulus

entity tb_penc8to3 port
end tb_penc8to3;

architecture arch_tb_penc8to3 of tb_penc8to3 is
signal i : in STD_LOGIC_VECTOR(7 downto 0) := ,00000000';
signal y : out STD_LOGIC_VECTOR(2 downto 0);

component penc8to3 port (
: in STD_LOGIC_VECTOR(7 downto 0);
: out STD_LOGIC_VECTOR(2 downto 0)); end
component; y

begin
DUT: penc8to3 port map (i, y);

process
begin
i = i + '1';
wait for 10ns;
end process End

architecture

Plot the wave form for all possible select lines
Synthesize the design
Elaborate the design and dump the bit file into FPGA

```

Try

1. Modify VHDL behavioral model with gate level model for 8 to 3 encoder. Plot the waveforms.
2. Realize 8to3 priority encoder using 2x1 mux and implement with VHDL gate level model.

12. Exercises on Shift Register

Shift registers, like counters, are a form of sequential logic. Sequential logic, unlike combinational logic, is not only affected by the present inputs but also by the prior history. In other words, sequential logic remembers past events. Shift registers produce a discrete delay of a digital signal or waveform.

12.1 4- bit barrel shifter

A barrel shifter is a digital circuit that can shift a data word by a specified number of bits without use of any sequential logic, only pure combinational logic. There are 3 type of bitwise shift operation: logical shift, arithmetic shift, and circular shift (rotate). The data can be shifted to the left as well as to the right circular shift.

Hints

Consider the port

specification as Input

datain // 8 bit wide

Input sel // 3 bit wide to support 16

operations Input dataout // 8 bit

wide

The circuit allows rotating the input data word right, where the amount of rotating is selected by the control inputs. The circuit can design by three stages of 2:1 multiplexer.

When all multiplexer select inputs are active (low), the input data passes straight through the cascade of the multiplexers and the output data ($q_7.....q_0$) is equal to the input data ($d_7.....d_0$). When S_2 control signal is selected, the first stage of multiplexers performs a rotate-right by one bit operation, due to their inter-connection to the next lower input.

The second stage of multiplexers performs a rotate-right by two bits when S_1 control signal is selected. Here the corresponding multiplexer inputs are connected to their second next-lower input.

Try:

1. Modify the barrel shifter by changing the number of select lines to 4 bit wide to support 16 different functionalities.
2. Modify the barrel shifter by changing the number of data lines to 16 bit wide to support 16 different functionalities.

12.2. 8- bit ALU

Arithmetic Logic Unit (ALU) is one of the most important digital logic components in CPUs. It normally executes arithmetic operations such as addition, subtraction, multiplication, division, etc. and logic operations such as and, or, xor, xnor, nand, nor, not, buffer, rotate and shift operations.

Hints

Consider the port

specification as Input A, B

// 8 bit wide

Input sel // 4 bit wide to support 16

operations Input dataout // 8 bit

wide

// The logic and arithmetic operations being implemented in the

ALU are as follows: Arithmetic Addition: $ALU_Out = A + B$;

Arithmetic Subtraction $ALU_Out = A$

- B; Arithmetic Multiplication ALU_Out

= $A * B$; Arithmetic Division $ALU_Out = A$

/ B;

Logical Shift Left $ALU_Out = A$ logical shifted

left by 1; Logical Shift Right $ALU_Out = A$ logical

shifted right by 1; Rotate Left $ALU_Out = A$ rotated

left by 1;

Rotate Right $ALU_Out = A$ rotated right by 1;

Logical AND $ALU_Out = A \text{ AND } B$;

Logical OR $ALU_Out = A \text{ OR } B$;

Logical XOR $ALU_Out = A \text{ XOR } B$;

Logical NOR $ALU_Out = A \text{ NOR } B$;

Logical NAND $ALU_Out = A \text{ NAND } B$;

Logical XNOR $ALU_Out = A \text{ XNOR } B$;

Greater comparison $ALU_Out = 1$ if $A > B$

else 0; Equal comparison

$ALU_Out = 1$ if $A = B$ else 0;

The second stage of multiplexers performs a rotate-right by two bits when S_1 control signal is selected. Here the corresponding multiplexer inputs are connected to their second next-lower input.

Try:

1. Modify the ALU by changing the number of select lines to 4 bit wide to support 16 different functionalities.
2. Modify the ALU by changing the number of data lines to 16 bit wide to support 16 different functionalities.

13 Exercises on Latches and Flip-flops

Computers and calculators use Flip-flop for their memory. A combination of number of flip flops will produce some amount of memory. Flip flop is formed using logic gates, which are in turn made of transistors. Flip flop are basic building blocks in the memory of electronic devices. Each flip flop can store one bit of data. Latches and flip – flops are both 1 – bit binary data storage devices. The main difference between a latch and a flip – flop is the triggering mechanism. Latches are transparent when enabled, whereas flip – flops are dependent on the transition of the clock signal i.e. either positive edge or negative edge.

13.1 SR latch, JK latch, D latch and T latch

Construct an SR latch using NOR gates. Verify its operation and demonstrate the circuit. Write an entity declaration for a positive level-triggered SR-latch with asynchronous active-low preset and clear inputs, and Q and outputs. Include concurrent assertion statements and passive processes as necessary in the entity declaration to verify that,

- The preset and clear inputs are not activated simultaneously,
- The setup time of 6 ns from the J and K inputs to the rising clock edge is observed,
- The hold time of 2 ns for the J and K inputs after the rising clock edge is observed and
- The minimum pulse width of 5 ns on each of the clock, preset and clear inputs is observed.

Write a gate level architecture body for the SR latch and a test bench that exercises the statements in the entity declaration.

Hints

```
// Declare port signal

Input rst_l, clk;
Input s, r;

Output q, qb;

// Component declaration of NAND gates
Component NAND_gate (input a, b; output y);

// component instance in the architecture body of VHDL program
NAND_gate port map(s, qb, q);
NAND_gate port map(r, q, qb);

// Test bench for SR latch
Design a test bench to provide the stimulus for the inputs s, r
Generate a clk signal with 5MHz frequency
Generate the reset logic to reset the latch at initial
```

Try

1. Construct SR latch using NAND gates. Verify its operation and demonstrate the circuit
2. Construct JK latch using NAND gates. Verify its operation and demonstrate the circuit
3. Construct D latch using NAND gates. Verify its operation and demonstrate the circuit
4. Construct T latch using NAND gates. Verify its operation and demonstrate the circuit

13.2 JK flip-flop, D flip-flop and T flip-flop

Construct flip-flops using latches and verify its operation and demonstrate the circuit.

Write an entity declaration for a positive edge triggered jk ff with asynchronous active low preset and clear inputs. and Q and outputs.

Include concurrent assertion statements and passive processes as necessary in the entity declaration to verify that,

- The preset and clear inputs are not activated simultaneously,
- The setup time of 6 ns from the J and K inputs to the rising clock edge is observed,
- The hold time of 2 ns for the J and K inputs after the rising clock edge is observed
- The minimum pulse width of 5 ns on each of the clock, preset and clear inputs is observed.

Write a structural architecture body for the flipflop and a test bench that exercises the statements in the entity declaration.

Hints

The pseudo code for SR latch

```
// Declare port signal

Input rst_l, clk;
Input s, r;

Output q, qb;

// Component declaration of NAND gates
Component NAND_gate (input a, b; output y);

// component instance in the architecture body of VHDL program
NAND_gate port map(s, qb, q);
NAND_gate port map(r, q, qb);

// Test bench for SR latch
Design a test bench to provide the stimulus for the inputs s, r
Generate a clk signal with 5MHz frequency
```

Try

Design 2-bit register

1. Write component instantiation statements to model the structure shown by the schematic diagram in Figure. Assume that the entity `t1_74x74` and the corresponding architecture `basic` have been analyzed into the library work.

Exercises on Synchronous and Asynchronous Counters

A special type of sequential circuit used to count the pulse is known as a counter, or a collection of flip flops where the clock signal is applied is known as counters. The counter is one of the widest applications of the flip flop. Based on the clock pulse, the output of the counter contains a predefined state. The number of the pulse can be counted using the output of the counter.

There are two types of counters: Synchronous counter and Asynchronous counter.

14.1: 4-bit synchronous counter with synchronous reset

Build an entity for a 4-bit counter with synchronous reset input. Include a process in the entity declaration that measures the duration of each reset pulse and reports the duration at the end of each pulse.

Hints

```
/** Declare the port signals */
entity counter_synrst port (
clk, rst: in std_logic;
q: out std_logic_vector(3 downto 0)); end
counter_synrst;

/** architecturebody */
architecture arch_counter_synrst of counter_synrst is

Begin
Process (clk, rst) Begin

If clk'event and clk = '1' then If rst
then
q = ,0000';
else
q = q + 1;
end if;
end process

End arch_counter_synrst;

//Write the test bench for providing the stimulus

entity tb_counter_synrst port
end tb_counter_synrst;

architecture arch_tb_counter_synrst of tb_counter_synrst is

signal clk, rst: std_logic := '0';
signal q: std_logic_vector(3 downto 0);

component counter_synrst port ( clk,
rst: in std_logic;
q: out std_logic_vector(3 downto 0)); end
component;

begin
```

```

DUT: counter_synrst port map (gray_code, binary_code);

Process
begin
clk = ~clk; Wait for 10
ns;
end process;

Process
begin
rst = '0'; Wait for 10
ns; rst = '1'; wait;
end process; end

architecture

// After post simulation

Simulate the design using Xilinx software

Plot the wave forms and verify the functionality of the design

Synthesize the design

```

Try

- 1 Design 4-bit synchronous counter with asynchronous reset
- 2 Design 4-bit asynchronous counter with synchronous reset
- 3 Design 4-bit asynchronous counter with asynchronous reset

14.2 Decade counter with asynchronous reset

Construct an entity for a decade counter with asynchronous reset input. Include a process in the entity declaration that measures the duration of each reset pulse and reports the duration at the end of each pulse.

Hints

```

/**    Declare the port signals */
entity dec_counter_asynrst port (
    clk, rst: in std_logic;
q: out std_logic_vector(3 downto 0)); end
dec_counter_asynrst;

/**    architecturebody */
architecture arch_dec_counter_asynrst of dec_counter_asynrst is

Begin
Process (clk, rst) Begin

If rst then
q = ,0000';
elif clk'event and clk = '1' then

```

```
if (q = ,1010') then
q = ,0000';
else
q = q + 1;
end if;
```

```
end if; end process
```

```
End arch_dec_counter_asynrst;
```

```
//Write the test bench for providing the stimulus entity
```

```
tb_dec_counter_asynrst port
end tb_dec_counter_asynrst;
```

```
architecture arch_tb_dec_counter_asynrst of tb_dec_counter_asynrst is
```

```
signal clk, rst: std_logic := '0';
signal q: std_logic_vector(3 downto 0);
```

```
component dec_counter_asynrst port ( clk, rst: in
std_logic;
q: out std_logic_vector(3 downto 0)); end component;
```

```
begin
```

```
DUT: dec_counter_asynrst port map (clk, rst, q);
```

```
Process begin
clk = ~clk; Wait for 10 ns;
end process;
```

```
Process begin
rst = '0'; Wait for 10 ns; rst = '1';
wait;
end process; end
```

```
architecture
```

```
// After post simulation
```

Simulate the design using Xilinx software

Plot the wave forms and verify the functionality of the design Synthesize the design
Elaborate the design and create bit file Dump the bit file
in zybo fpga

Try

- 1 Design decade synchronous counter with synchronous reset.
- 2 Design counter to count the events from 3 to 12.

14.2.1.1 4-bit serial in serial out shift register (SISO)

In digital systems it is often necessary to have circuits that can shift the bits of a vector by one or more bit positions to the left or right. Design a circuit that can shift a four-bit vector $W = w_3w_2w_1w_0$ one bit position to the right when a control signal Shift is equal to 1. Let the outputs of the circuit be a four-bit vector $Y = y_3y_2y_1y_0$ and a signal k , such that if Shift = 1 then $y_3 = 0$, $y_2 = w_3$, $y_1 = w_2$, $y_0 = w_1$, and $k = w_0$. If Shift = 0 then $Y = W$ and $k = 0$.

Build an entity for a shift register to drive the register serially and output the data serially. Write a test bench architecture to simulate and verify the design.

Hints

```
/**  Declare the port signals */
entity siso port (
  clk, rst: in std_logic; sin : in
  std_logic;
  q: out std_logic_vector(3 downto 0)); end siso;

/**  architecturebody */
architecture arch_asiso of siso is

Begin
Process (clk, rst) Begin

If rst then
q = ,0000';
elif clk'event and clk = '1' then q[3] =
sin;
q[2] = q[3];
q[1] = q[2];
q[0] = q[1];

end if; end
process

End arch_asiso;

//Write the test bench for providing the stimulus

entity tb_siso port
end tb_siso;

architecture arch_tb_asiso of tb_siso is

signal clk, rst: std_logic := '0';
signal sin: std_logic := '0';
signal q: std_logic_vector(3 downto 0);

component siso port (
  clk, rst: in std_logic;
```

```

sin : in std_logic;
q: out std_logic_vector(3 downto 0)); end
component;

begin

DUT: siso port map (clk, rst, sin, q);

Process
begin
clk = ~clk; Wait for 10
ns;
end process;

Process
begin
rst = '0';
Wait for 10 ns; rst =
'1'; wait;
end process;

Process
begin
sin = ~sin;
Wait for 25 ns;
end process;

end architecture

// After post simulation

Simulate the design using Xilinx software

Plot the wave forms and verify the functionality of the design

Synthesize the design
Elaborate the design and create bit file
Dump the bit file in zybo FPGA

```

Try

- 1 Design 4-bit serial in parallel out shift register (SIPO).
- 2 Design 4-bit parallel in serial out shift register (PISO).
- 3 Design 4-bit parallel in parallel out shift register (PIPO).

14.2.2 Introduction

A carry look-ahead adder (CLA) is an electronic adder used for binary addition. Due to the quick additions performed, it is also known as a fast adder. The CLA logic uses the concepts of generating and propagating carries. We can say that the CLA adder is the successor of the Ripple Carry Adder

14.2.2.1 Carry look ahead adder

Build 4- bit carry look ahead adder (CLA) and justify the speed of operation CLA is more than ripple carry adder

Develop a functional model of a 3-bit carry-look-ahead adder. The adder has two 3-bit data inputs, a (2 downto 0) and b(3 downto 0); a 3-bit data output, s(2 downto 0); a carry input, c_in; a carry output, c_out; a carry generate output, g; and a carry propagate output, p. The adder is described by the logic equations and associated propagation delays: where the G_i are the intermediate carry generate signals, the P_i are the intermediate carry propagate signals and the C_i are the intermediate carry signals. C_{-1} is c_{in} and C_3 is c_{out} . Your model should use the expanded equation to calculate the intermediate carries, which are then used to calculate the sums.

$$s_i = a_i \oplus b_i \oplus c_{i-1}$$

$$g_i = a_i . b_i$$

$$p_i = a_i + b_i$$

$$c_i = g_i + p_i c_{i-1}$$

Hints

```

/** Declare the port signals */
entity cla port (
a, b: in std_logic_vector(2 downto 0); sum: out
std_logic_vector(2 downto 0); cout: out
std_logic);
end cla;

/** architecturebody */
architecture arch_cla of cla is

Begin
g0 = a[0] & b[0];
p0 = a[0] | b[0]; c1 =
g0 + p0 & c0;
. . . . .
. . . . .
. . . . .

End arch_cla;

//Write the test bench for providing the stimulus
entity tb_cla port
end tb_cla;

architecture arch_tb_cla of tb_cla is

signal a, b: std_logic_vector(2 downto 0) := ,000';
signal cin: std_logic := '0';
signal sum: std_logic_vector(s downto 0);

component cla port (
a, b: in std_logic_vector(2 downto 0); sum: out
std_logic_vector(2 downto 0); cout: out
std_logic);
end component;

begin

DUT: cla port map (clk, rst, sin, q);

Process
begin

```



```

a = a + ,0110'
wait for 10 ns;
b = b + ,1010'
end process; end

architecture

// After post simulation

Simulate the design using Xilinx software

Plot the wave forms and verify the functionality of the design

Synthesize the design
Elaborate the design and create bit file
Dump the bit file in zybo FPGA

```

Try:

1. Design and implement 4-bit carry look ahead adder
2. Design and implement 4-bit ripple carry adder

V. TEXT BOOKS

1. Douglas Perry, “VHDL”, Tata McGraw Hill, 4th edition, 2002.
2. W.H. Gothmann, “Digital Electronics- An introduction to theory and practice”, PHI, 2nd edition, 2006.

VI. REFERENCE BOOKS

1. Jacob Millman, Herbert Taub, Mothiki S PrakashRao, “Pulse Digital and Switching Waveforms”, Tata McGraw-Hill, 3rd edition, 2008.
2. David A. Bell, “Solid State Pulse Circuits”, PHI, 4th edition, 2002.
3. D Roy Chowdhury, “Linear Integrated Circuits”, New Age International (p) Ltd, 2nd edition, 2003.
4. Ramakanth A. Gayakwad, “Op-Amps linear ICs”, PHI, 3rd edition, 2003.

VII. MATERIALS ONLINE

1. Course template
2. Lab Manual



INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal - 500 043, Hyderabad, Telangana

COURSE CONTENT

DevOps Engineering								
IV Semester: AERO ME CE EEE CSE IT CSE (AI&ML) CSE (DS) CSE (CS)								
Course Code	Category	Hours / Week			Credits	Maximum Marks		
ACSD18	Skill	L	T	P	C	CIA	SEE	Total
		2	0	0	0	40	60	100
Contact Classes: 32	Tutorial Classes: Nil	Practical Classes: Nil			Total Classes: 32			
Prerequisite: OBJECT ORIENTED PROGRAMMING								

I. COURSE OVERVIEW:

DevOps, a combination of "development" and "operations," is a software development methodology that emphasizes collaboration and communication between software developers and IT operations professionals. The goal of DevOps is to streamline the software delivery process, from code development to deployment and maintenance, by breaking down silos between development and operations teams.

II. COURSES OBJECTIVES:

The students will try to learn

- I. The DevOps Concepts for business cases, cloud provisioning and management services .
- II. The model canvas for DevOps use cases.
- III. The virtual machines and containers for designing of applications
- IV. The code with various aspects in continuous deployment / development.

III. COURSE OUTCOMES:

At the end of the course students should be able to:

- CO 1 Understands the DevOps concepts in continuous delivery / development of applications.
- CO 2 Create the DevOps applications using various tools and technologies.
- CO 3 Examine the virtual machines and containers for managing the files
- CO 4 Apply cloud services for deployment the applications in a real-time
- CO 5 Perform web security and testing the code with appropriate tools

IV. COURSE SYLLABUS:

MODULE 1: DevOps Concepts

Understanding DevOps movement, DevOps with changing time, The water fall model, Agile Model, Collaboration, Why DevOps, Benefits of DevOps, DevOps life cycle- all about continuous, Build Automation, Continuous Integration, Continuous Management, Continuous Delivery / Continuous Development, The agile wheel of wheels.

MODULE 2: DevOps Tools and Technologies

Code Repositories : Git, Differences between SVN and Git, Build tools – Maven, Continuous integration tools – Jenkins, Container Technology – Docker, Monitoring Tools, Continuous integration with Jenkins 2, Creating

built-in delivery pipelines, Creating Scripts, Creating a pipeline for compiling and executing test units, Using the Build Pipeline plugin, Integrating the deployment operation

MODULE 3: Docker Containers

Overview of Docker containers, Understanding the difference between virtual machines and containers, Installation and configuration of Docker, Creating your first Docker container, Managing containers, Creating a Docker image from Docker file, An overview of Docker's elements, Creating a Dockerfile, Writing a Dockerfile, Building and running a container on a local machine, Testing a container locally, Pushing an image to Docker Hub

MODULE 4: Cloud Provisioning and Configuration Management with Chef, Managing Containers Effectively with Kubernetes

Amazon EC2, Creating and configuring a virtual machine in Amazon Web Services, Prerequisite – deploying our application on a remote server, Deploying the application on AWS, Deploying the application in a Docker container.

Kubernetes architecture overview, Installing Kubernetes on a local machine, Installing the Kubernetes dashboard, Kubernetes application deployment, Using Azure Kubernetes Service (AKS), Creating an AKS service, Configuring kubectl for AKS, The build and push of the image in the Docker Hub

MODULE 5: Testing the Code

Manual testing, Unit testing, JUnit in general and JUnit in particular, A JUnit example, Automated integration testing, Docker in automated testing, Performance testing, Automated acceptance testing, Automated GUI testing, Integrating Selenium tests in Jenkins, JavaScript testing, Testing backend integration points, Test-driven development, A complete test automation scenario, Manually testing our web application.

V. TEXT BOOKS:

1. Mitesh Soni, “DevOps for Web Development”, Packt Publishing, 2016.
2. Mikael Krief, “Learning DevOps - The complete guide to accelerate collaboration with Jenkins, Kubernetes, Terraform and Azure DevOps”, Packt Publishing, 2019.

VI. REFERENCE BOOKS:

1. Joakim Verona, “Practical DevOps”, Packt Publishing, 2016.
2. Michael Huttermann, “DevOps for Developers”, A press publishers, 2012.
3. Sanjeev Sharma, “The DevOps Adoption Playbook”, Published by John Wiley & Sons, Inc.2017.
4. Sanjeev Sharma & Bernie Coyne, “DevOps for Dummies”, Published by John Wiley & Sons, Inc.

VII. REFERENCE BOOKS:

1. <https://www.geeksforgeeks.org/devops-tutorial/>
2. <https://www.javatpoint.com/devops>
3. <https://azure.microsoft.com/en-in/solutions/devops/tutorial>
4. <https://www.guru99.com/devops-tutorial.html>



INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal - 500 043, Hyderabad, Telangana

UNDERTAKING BY STUDENT / PARENT

“To make the students attend the classes regularly from the first day of starting of classes and be aware of the College regulations, the following undertaking form is introduced which should be signed by both student and parent. The same should be submitted to the Dean of Academic”.

I, Mr. / Ms. ----- joining I Semester / III Semester for the academic year 20 - 20 / 20 - 20 in Institute of Aeronautical Engineering, Hyderabad, do hereby undertake and abide by the following terms, and I will bring the ACKNOWLEDGEMENT duly signed by me and my parent and submit it to the Dean of Academic.

1. I will attend all the classes as per the timetable from the starting day of the semester specified in the institute Academic Calendar. In case, I do not turn up even after two weeks of starting of classes, I shall be ineligible to continue for the current academic year.
2. I will be regular and punctual to all the classes (theory/laboratory/project) and secure attendance of not less than 75% in every course as stipulated by Institute. I am fully aware that an attendance of less than 65% in more than 60% of theory courses in a semester will make me lose one year.
3. I will compulsorily follow the dress code prescribed by the college.
4. I will conduct myself in a highly disciplined and decent manner both inside the classroom and on campus, failing which suitable action may be taken against me as per the rules and regulations of the institute.
5. I will concentrate on my studies without wasting time in the Campus/Hostel/Residence and attend all the tests to secure more than the minimum prescribed Class/Sessional Marks in each course. I will submit the assignments given in time to improve my performance.
6. I will not use Mobile Phone in the institute premises and also, I will not involve in any form of ragging inside or outside the campus. I am fully aware that using mobile phone to the institute premises is not permissible and involving in Ragging is an offence and punishable as per JNTUH/UGC rules and the law.
7. I declare that I shall not indulge in ragging, eve-teasing, smoking, consuming alcohol drug abuse or any other anti-social activity in the college premises, hostel, on educational tours, industrial visits or elsewhere.
8. I will pay tuition fees, examination fees and any other dues within the stipulated time as required by the Institution / authorities, failing which I will not be permitted to attend the classes.
9. I will not cause or involve in any sort of violence or disturbance both within and outside the college campus.
10. If I absent myself continuously for 3 days, my parents will have to meet the HOD concerned / Principal.
11. I hereby acknowledge that I have received a copy of BT23 Academic Rules and Regulations, course catalogue and syllabus copy and hence, I shall abide by all the rules specified in it.

ACKNOWLEDGEMENT

I have carefully gone through the terms of the undertaking mentioned above and I understand that following these are for my/his/her own benefit and improvement. I also understand that if I/he/she fail to comply with these terms, shall be liable for suitable action as per Institute/JNTUH/AICTE/UGC rules and the law. I undertake that I/he/she will strictly follow the above terms.

Signature of Student with Date

Signature of Parent with Date
Name & Address with Phone Number