INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)
Dundigal, Hyderabad -500 043

Computer Science and Engineering

COURSE INFORMATION SHEET

Course Title	**COMPUTER PROGRAMMING (Common for AE/ CE/ ME)**			
Course Code	ACS001			
Programme	B.Tech			
Semester	II			
Course Type	Foundation			
Regulation	**IARE-R16**			
Course Structure	**Lectures**	**Tutorials**	**Practicals**	**Credits**
	3	-	-	3
Course Coordinator	Mr.N Ramanjaneya Reddy Associate Professor,CSE Dept.			
Course Faculty	Mr.N Ramanjaneya Reddy, Mr. N Poorna Chandra Rao, Mr.S Lakshman Kumar, Ms.A Uma Datta, Ms.A Swapna, Ms.A Lakshmi,			

I. COURSE OVERVIEW:

The course covers the basics of programming and demonstrates fundamental programming techniques, customs and terms including the most common library functions and the usage of the preprocessor. This course helps the students in gaining the knowledge to write simple C language applications, mathematical and engineering problems. This course helps to undertake future courses that assume this programming language as a background in computer programming. Topics include variables, data types, functions, control structures, pointers, strings, arrays and dynamic allocation principles. This course in reached to student by power point presentations, lecture notes, and lab involve the problem solving in mathematical and engineering areas.

II. COURSE PRE-REQUISITES:

Level	Course Code	Semester	Prerequisites	Credits
-	-	-	Basic Mathematics and Logical Thinking	----

III. MARKS DISTRIBUTION

Subject	SEE Examination	CIA Examination	Total Marks
Computer Programming	70 Marks	30 Marks	100

Semester End Examination (SEE):
The SEE is conducted for 70 marks of 3 hours duration. The syllabus for the theory courses is divided into FIVE units and each unit carries equal weightage in terms of marks distribution. The question paper pattern is as follows: two full questions with ‘either’ ‘or’ choice will be drawn from each unit. Each question carries 14 marks.

Continuous Internal Assessment (CIA):
CIA is conducted for a total of 30 marks, with 25 marks for Continuous Internal Examination (CIE) and 05 marks for Quiz / Alternative Assessment Tool (AAT).

Continuous Internal Examination (CIE):
The CIE exam is conducted for 25 marks of 2 hours duration consisting of two parts. Part–A shall have five compulsory questions of one mark each. In part–B, four out of five questions have to be answered where, each question carries 5 marks. Marks are awarded by taking average of marks scored in two CIE exams.

Quiz / Alternative Assessment Tool (AAT):
Two Quiz exams shall be online examination consisting of 20 multiple choice questions and are be answered by choosing the correct answer from a given set of choices (commonly four). Marks shall be awarded considering the average of two quizzes for every course. The AAT may include seminars, assignments, term paper, open ended experiments, micro projects, five minutes video and MOOCs.

IV. DELIVERY / INSTRUCTIONAL METHODOLOGIES:

√	CHALK & TALK	√	QUIZ	√	ASSIGNMENTS	√	MOOCs
√	LCD / PPT	√	SEMINARS	√	MINI PROJECT	√	VIDEOS
√	OPEN ENDED EXPERIMENTS						

V. ASSESSMENT METHODOLOGIES – DIRECT

√	CIE EXAMS	√	SEE EXAMS	√	ASSIGNEMNTS	√	SEMINARS
√	LABORATORY PRACTICES	√	STUDENT VIVA	√	MINI PROJECT	X	CERTIFICATION
√	TERM PAPER						

VI. ASSESSMENT METHODOLOGIES – INDIRECT

√	ASSESSMENT OF COURSE OUTCOMES (BY FEEDBACK, ONCE)	√	STUDENT FEEDBACK ON FACULTY (TWICE)
X	ASSESSMENT OF MINI PROJECTS BY EXPERTS		

VII. COURSE OBJECTIVES:

The course should enable the students to:

I. **Learn** adequate knowledge by problem solving techniques.
II. **Understand** programming skills using the fundamentals and basics of C Language.
III. **Improve** problem solving skills using arrays, strings, and functions.
IV. **Understand** the dynamics of memory by pointers.
V. **Study** files creation process with access permissions.

VIII. COURSE LEARNING OUTCOMES:

Students, who complete the course, will have demonstrated the ability to do the following:

CACS001.01	Identify and understand the working of key components of a computer system.
CACS001.02	Analyze a given problem and develop an algorithm to solve the problem.
CACS001.03	Describe the fundamental programming constructs and articulate how they are used to develop a program with a desired runtime execution flow.
CACS001.04	Gain knowledge to identify appropriate C language constructs to write basic programs.
CACS001.05	Identify the right data representation formats based on the requirements of the problem.
CACS001.06	Describe the operators, their precedence and associativity while evaluating expressions in program statements..
CACS001.07	Understand branching statements, loop statements and use them in problem solving.
CACS001.08	Learn homogenous derived data types and use them to solve statistical problems.
CACS001.09	Understand procedural oriented programming using functions.
CACS001.10	Understand how recursion works and write programs using recursion to solve problems.
CACS001.11	Differentiate call by value and call by reference parameter passing mechanisms.
CACS001.12	Understand pointers conceptually and apply them in C programs.
CACS001.13	Distinguish homogenous and heterogeneous data types and apply them in solving data processing applications.
CACS001.14	Explain the concept of file system for handling data storage and apply it for solving problems.
CACS001.15	Differentiate text files and binary files and write the simple C programs using file handling functions.
CACS001.16	Apply the concepts to solve real-time applications using the features of C language.
CACS001.17	Possess the knowledge and skills for employability and to succeed in national and international level competitive examinations.

IX. HOW PROGRAM OUTCOMES ARE ASSESSED:

Program Outcomes		Level	Proficiency assessed by
PO 1	**Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.	H	Assignments/Quiz
PO 2	**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.	H	Assignments/Quiz
PO 3	**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.	S	Mini Project
PO 4	**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.	S	Open ended experiments /
PO 5	**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.	S	Mini Project

Program Outcomes		Level	Proficiency assessed by
PO 6	**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.	N	---
PO 7	**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.	N	---.
PO 8	**Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.	N	---
PO 9	**Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.	N	---
PO 10	**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.	S	Seminars / Term Paper / 5 minutes video
PO 11	**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.	N	---
PO 12	**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.	S	-----

N= None S= Supportive H = Highly Related

X. HOW PROGRAM SPECIFIC OUTCOMES ARE ASSESSED:

Program Specific Outcomes		Level	Proficiency assessed by
PSO 1	**Engineering Knowledge:** Graduates shall demonstrate sound knowledge in analysis, design, laboratory investigations and construction aspects of civil engineering infrastructure, along with good foundation in mathematics, basic sciences and technical communication.	H	Lectures, Assignments
PSO 2	**Broadness and Diversity:** Graduates will have a broad understanding of economical, environmental, societal, health and safety factors involved in infrastructural development, and shall demonstrate ability to function within multidisciplinary teams with competence in modern tool usage.	H	Mini Projects
PSO 3	**Self-Learning and Service:** Graduates will be motivated for continuous self-learning in engineering practice and/ or pursue research in advanced areas of civil engineering in order to offer engineering services to the society, ethically and responsibly.	S	Guest Lectures

N - None S - Supportive H - Highly Related

XI. SYLLABUS:

UNIT I
Introduction to computers: Computer systems, computing environments, computer languages, creating and running programs algorithms, flowcharts; Introduction to C language: History of C, basic structure of C programs, process of compiling and running a C program, C tokens, keywords, identifiers, constants, strings, special symbols, variables, data types; Operators and expressions: Operators, arithmetic, relational and logical, assignment operators, increment and decrement operators, bitwise and conditional operators, special operators, operator precedence and associativity, evaluation of expressions, type conversions in expressions, formatted input and output.

UNIT II
Control structures: Decision statements; if and switch statement; Loop control statements: while, for and do while loops, jump statements, break, continue, goto statements; Arrays: Concepts, one dimensional arrays, declaration and initialization of one dimensional arrays, two dimensional arrays, initialization and accessing, multi dimensional arrays; Strings concepts: String handling functions, array of strings.
UNIT III
Functions: Need for user defined functions, function declaration, function prototype, category of functions, inter function communication, function calls, parameter passing mechanisms, recursion, passing arrays to functions, passing strings to functions, storage classes, preprocessor directives. **Pointers:** Pointer basics, pointer arithmetic, pointers to pointers, generic pointers, array of pointers, pointers and arrays, pointers as functions arguments, functions returning pointers.
UNIT IV
Structures and unions: Structure definition, initialization, accessing structures, nested structures, arrays of structures, structures and functions, passing structures through pointers, self referential structures, unions, bit fields, typedef, enumerations; Dynamic memory allocation: Basic concepts, library functions.
UNIT V
Files: Streams, basic file operations, file types, file opening modes, file input and output functions, file status functions, file positioning functions, command line arguments.

TEXT BOOKS:

1	Stephen G. Kochan, "Programming in C", Addison - Wesley Professional, 4th Edition, 2014.
2	B.A. Forouzan and R.F. Gilberg, "Computer Science: A Structured Programming Approach Using C", 3e, Cengage Learning. .

REFERENCES:

1	W. Kernighan Brian, Dennis M. Ritchie, "The C Programming Language", PHI Learning, 2nd Edition, 1988.
2	Yashavant Kanetkar, "Exploring C", BPB Publishers, 2nd Edition, 2003.
3	E. Balagurusamy, "Programming in ANSI C", McGraw Hill Education, 6th Edition, 2012
4	Schildt Herbert, "C: The Complete Reference", Tata McGraw Hill Education, 4th Edition, 2014.
5	R. S. Bichkar, "Programming with C", Universities Press, 2nd Edition, 2012.
6	Dey Pradeep, Manas Ghosh, "Computer Fundamentals and Programming in C", Oxford University Press, 2nd Edition, 2006.

XII. COURSE PLAN:

The course plan is meant as a guideline. There may probably be changes.

Lecture No	Topic Outcomes	Topic/s to be covered	Reference
1 – 2	**Identify** basic parts of computers.	Introduction to Computers: computer systems, computing environments.	T2:1.1-1.2
3 – 4	**Understand** the basics of a programming language.	Computer languages, creating and running programs, program development.	T2:2.1-2.2
5 – 6	**Develop** algorithms and flowcharts for given problem.	Algorithms, flowcharts	T2:1.4-1.5
7 – 8	**Understand** the basics of a C programming.	Introduction to C Language: background, C programs.	T2:2.1-2.2
9 – 10	**Understand** the basic building blocks of a C program.	Identifiers, data types, Input/output, variables, constants, Operators (arithmetic, relational, logical, bitwise etc).	T2: 2.3-2.6,7
11 – 12	**Understand** the rules of precedence and associativity in evaluating expressions.	Expressions, precedence and associativity, expression evaluation, type conversions.	T2:3.1-3.5

Lecture No	Topic Outcomes	Topic/s to be covered	Reference
13 – 14	**Write** programs using two-way and multi-way selection.	Statements - selection statements (making decisions) – if statement, switch statement.	T2: 5.2-5.3
15 – 16	**Identify** appropriate loop construct for a given problem.	Repetition statement (loops)-while, do-while statements, for statements, loop examples.	T2: 6.1-6.6
17 – 18	**Identify** appropriate loop construct for a given problem.	Other statements related to looping – break, continue, go to, simple C program examples.	T2: 6.7
19 – 20	**Understand** the usage of arrays to store homogenous data. Understand the applications of arrays	Arrays- Concepts, using arrays in C, declaration and initialization of one dimensional array, C program example. Two dimensional arrays, initialization and accessing, multi dimensional arrays, C program example.	T2: 8.1-8.3,8.7-8.8
21 – 22	**Write** programs to manipulate strings.	Strings – Strings concepts: String handling functions, array of strings, C program examples.	T2: 11.1-11.5
23-- 24	**Design** and **implement** multi-function programs.	Functions- Need for user defined functions, function declaration, function prototype. Category of functions, inter function communication, function calls, parameter passing mechanisms.	T2: 4.1-4.5
25	**Implement** recursive solutions to problems and passing arrays and strings as parameters and differentiate local and global scope of variables and write programs using preprocessor directives.	Recursion, passing arrays to functions, passing strings to functions, Storage classes and preprocessor commands.	T1:7 T2: 6.9 T2:G.1
26 -- 27	**Understand** the basic concepts of pointers. Write effective programs using pointers.	Pointer basics, pointer arithmetic, pointers to pointers, generic pointers. Pointer applications-Arrays and pointers, pointer arithmetic and arrays, passing an array to a function.	T1:10 T2:10.1-10.2
28 – 29	**Write** effective programs using pointers.	Array of pointers, pointers and arrays, pointers as functions arguments, functions returning pointers	T2:10.3-10.5
30 – 31	**Understand** the usage and applications of structures to store heterogeneous data.	Structures – declaration, initialization, accessing structures, operations on structures.	T1:8
32 – 33	**Understand** the usage and applications of structures to store heterogeneous data.	Complex structures, structures and functions, passing structures through pointers, self-referential structures.	T2: 12.3-12.4
34 – 35	**Differentiate** structures and unions in terms of memory allocation and understand, bit fields, the enumerated data types.	Unions, C programming examples, Bit fields, typedef, enumerations.	T2:12.4 T2:12.1-12.2
36 -- 38	**Write** effective programs using pointers for dynamic memory allocation.	Dynamic memory allocation: Basic concepts, library functions.	T2:10.4
39 – 40	**Understand** the basic properties and characteristics of files.	**Files:** Concept of a file, streams, types of files and file opening modes.	R3:12.1-12.3
41 – 42	**Use** library file I/O functions for storing data on secondary storage.	File input/output functions (standard input/output functions for files).	R3:12.4
43 – 44	**Write** programs to handle simple file I/O errors.	File status functions (error handling), positioning functions, C program examples.	R3:12.5
45	**Understand** the use of command-line arguments.	Command-line arguments.	R3:12.7

XIII. GAPS IN THE SYLLABUS - TO MEET INDUSTRY / PROFESSION REQUIREMENTS:

S NO	DESCRIPTION	PROPOSED ACTIONS	RELEVANCE WITH POs	RELEVANCE WITH PSOs
1	Updating latest version and new features of the C language	Laboratory Sessions	PO5	PSO2
2	Familiarizing the role of C language in developing system level programs.	Assignments/ Industrial visits	PO1,PO2	PSO2
3	Familiarizing different areas where C language can be used.	Seminars	PO12	PSO3
4	Solving different problems and Practicing various debugging strategies to become a good programmer	Extra Lab Sessions, Participating in Coding contests.	PO2	PSO3

XIV. MAPPING COURSE OBJECTIVES LEADING TO THE ACHIEVEMENT OF PROGRAM OUTCOMES AND PROGRAM SPECIFIC OUTCOMES:

Course Objectives (COs)	Program Outcomes (POs)												Program Specific Outcomes (PSOs)		
	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
I	H	H										S	H	H	
II		H	S		S								S	H	
III	S	H	S										H		S
IV	H	S											H	S	
V	H	S											H	S	S

S= Supportive **H = Highly Related**

XV. MAPPING COURSE LEARNING OUTCOMES LEADING TO THE ACHIEVEMENT OF PROGRAM OUTCOMES AND PROGRAM SPECIFIC OUTCOMES:

Course Learning Outcomes (CLOs)	Program Outcomes (POs)												Program Specific Outcomes (PSOs)		
	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CACS001.01	H			S											
CACS001.02		H	S									S	H	S	
CACS001.03		H			S								H		S
CACS001.04	H			S								S	S		
CACS001.05		H	S		S							S	S	S	
CACS001.06	H	S		S									H		S
CACS001.07	H				S							S	S		
CACS001.08	S	H	S	S									H	S	

CACS001.09	S	H			S							S	H		S
CACS001.10	S	H											S	S	
CACS001.11	H	S										S		H	
CACS001.12	H												S		H
CACS001.13	S	H			S							S	H		S
CACS001.14	S	H			S							S	H		S
CACS001.15	S	H			S							S	H		S
CACS001.16	S	H			S							H		H	H
CACS001.17	S	H			S							H		H	H

S= Supportive **H = Highly Related**

XVI. DESIGN BASED PROBLEMS (DP) / OPEN ENDED PROBLEM:

1. Develop simple character-based Chess-game supporting standard partial chess moves. Chess board should be 8x8 Cell Board having each Cell of 4 characters. Basic chess board with empty shell should have W... Cell and B... For Black Cell. Wherever any player's Game elements such as Rook or Camel or King or Queen is on board Cell then it. Then it should be displayed such as BQN2 or WQN1 which indicated such as Queen of player-2 on black cell or queen of player-1 on white cell. Or Student can use his own conventions. Student should be able to demonstrate 5 moves for each player minimum.

2. (Airline Reservations System) A small airline has just purchased a computer for its new automated reservations system. The president has asked you to program the new system. You are to write a program to assign seats on each flight of the airline's only plane (capacity: 10 seats).
   Your program should display the following menu of alternatives:
   Please type 1 for "first class"
   Please type 2 for "economy"
   If the person types 1, then your program should assign a seat in the first class section (seats 1-5). If the person types 2, then your program should assign a seat in the economy section (seats 6-10). Your program should then print a boarding pass indicating the person's seat number and whether it is in the first class or economy section of the plane.
   Use a single-subscripted array to represent the seating chart of the plane. Initialize all the elements of the array to 0 to indicate that all seats are empty. As each seat is assigned, set the corresponding elements of the array to 1 to indicate that the seat is no longer available.
   Your program should, of course, never assign a seat that has already been assigned. When the first class section is full, your program should ask the person if it is acceptable to be placed in the economy section (and vice versa). If yes, then make the appropriate seat assignment. If no, then print the message "Next flight leaves in 3 hours.

3. (*Maze Traversal*) The following grid is a double-subscripted array representation of a maze.

```
# # # # # # # # # # # #
# . . . # . . . . . . #
. . # . # . # # # # . #
# # # . # . . . . # . #
# . . . . # # # . # . .
# # # # . # . # . # . #
# . . # . # . # . # . #
# # . # . # . # . # . #
# . . . . . . . . # . #
# # # # # # . # # # . #
# . . . . . . # . . . #
# # # # # # # # # # # #
```

The # symbols represent the walls of the maze, and the periods (.) represent squares in the possible paths through the maze. There is a simple algorithm for walking through a maze that guarantees finding the exit (assuming there is an exit). If there is not an exit, you will arrive at the starting location again. Place your right hand on the wall to your right and begin walking forward. Never remove your hand from the wall. If the maze turns to the right, you follow the wall to the right. As long as you do not remove your hand from the wall, eventually you will arrive at the exit of the maze. There may be a shorter path than the one you have taken, but you are guaranteed to get out of the maze. Write recursive function mazeTraverse to walk through the maze. The function should receive as arguments a 12-by-12

character array representing the maze and the starting location of the maze. As mazeTraverse attempts to locate the exit from the maze, it should place the character X in each square in the path. The function should display the maze after each move so the user can watch as the maze is solved.

Prepared By:
Mr.N Ramanjaneya Reddy Associate Professor, CSE Dept.

HOD, CIVIL ENGINEERING