OPTIMIZATION TECHNIQUES LABORATORY

LAB MANUAL

Course Code	**:**	**AMEB15**
Regulations	**:**	**IARE-R18**
Class	**:**	**II Year II Semester(ME)**

Prepared by

Mrs. T.Vanaja, Assistant Professor.

DEPARTMENT OF MECHANICAL ENGINEERING

INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal – 500 043, Hyderabad

INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)
Dundigal, Hyderabad - 500 043
MECHANICAL ENGINEERING

Program Outcomes	
PO1	**Engineering knowledge**: Capability to apply the knowledge of Mathematics, science and Engineering in the field of Mechanical Engineering.
PO2	**Problem analysis:** An Ability to analyze complex engineering problems to arrive at relevant conclusions using knowledge of Mathematics, Science and Engineering.
PO3	**Design/development of solutions**: Competence to design a system, component or process to meet societal needs within realistic constraints.
PO4	**Conduct investigations of complex problems**: To design and conduct research oriented experiments as well as to analyze and implement data using research methodologies.
PO5	**Modern tool usage**: An ability to formulate solve complex engineering problem using modern engineering and Information technology tools.
PO6	**The engineer and society**: To utilize the engineering practices, techniques, skills to meet needs of the health, safety, legal, cultural and societal issues.
PO7	**Environment and sustainability**: To understand impact of engineering solutions in the societal context and demonstrate the knowledge for sustainable development.
PO8	**Ethics**: An understanding and implementation of professional and Ethical responsibilities.
PO9	**Individual and team work**: To function as an effective individual and as a member or leader in Multi-disciplinary environment and adopt in diverse teams.
PO10	**Communication**: An ability to assimilate, comprehends, communicate, give and receive instructions to present effectively with engineering community and society.
PO11	**Project management and finance**: An ability to provide leadership in managing complex engineering projects at multi-disciplinary environment and to become a professional engineer.
PO12	**Life-long learning**: Recognition of the need and an ability to engage in lifelong learning to keep abreast with technological changes.
Program Specific Outcomes	
PSO1	**Professional Skills:** To produce engineering professional capable of synthesizing and analyzing mechanical systems including allied engineering streams.
PSO2	**Modelling and Simulation Practices:** An ability to adopt and integrate current technologies in the design and manufacturing domain to enhance the employability.
PSO3	**Successful Career and Entrepreneurship:** To build the nation, by imparting technological inputs and managerial skills to become Technocrats.

INDEX

ATTAINMENT OF PROGRAM OUTCOMES & PROGRAM SPECIFIC OUTCOMES

Exp. No.	Experiment	Program Outcomes Attained	Program Specific Outcomes Attained
1	Matrix Operations	PO1, PO2	PSO1, PSO2
2	Matrix Operations	PO1, PO3	PSO1
3	Minimum Cost Path	PO1, PO2	PSO1, PSO2
4	Finding Maximum Number In An Array	PO1, PO4	PSO1, PSO2
5	Array Sorting	PO1, PO2, PO3	PSO2
6	Linear Programming Problem	PO1, PO2, PO3	PSO1
7	Queuing Problem	PO1, PO2	PSO1, PSO2
8	Sequencing Problem	PO1, PO3, PO4	PSO2
9	Game Theory	PO1, PO3, PO4	PSO1, PSO2
10	Assignment Problem	PO1, PO2	PSO1, PSO2
11	Dynamic Programming Problem	PO1, PO2, PO4	PSO1
12	Inventory Problem	PO1, PO2	PSO2, PSO3
13	Examinations	PO1, PO3	PSO2, PSO3

INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal - 500 043, Hyderabad

Mechanical Engineering Department

OPTIMIZATION TECHNIQUES LABORATORY

Course Overview:

Introduction to optimization techniques using both linear and non-linear programming. The focus of the course is on convex optimization though some techniques will be covered for non-convex function optimization too. After an adequate introduction to linear algebra and probability theory, students will learn to frame engineering minima maxima problems in the framework of optimization problems.

Course Out Comes:

CO 1: Understand the basic principles and concepts of Python

CO 2: Explore the applicability of programming skills in Python.

CO 3: Summarize various optimization techniques like LPP models .

CO 4: Analyze the transportation, inventory and assignment problems.

CO 5: Explain the concepts of sequencing, game theory and dynamic programming.

INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal - 500 043, Hyderabad

MECHANICAL ENGINEERING DEPARTMENT

INSTRUCTIONS TO THE STUDENTS

1. Students are required to attend alllabs.
2. Students should work individually in the hardware and softwarelaboratories.
3. Students have to bring the lab manual cum observation book, record etc along with them whenever they come for lab work.
4. Should take only the lab manual, calculator (if needed) and a pen or pencil to the workarea.
5. Should learn the prelab questions. Read through the lab experiment to familiarize themselves with the components and assemblysequence.
6. Should utilize 3 hour's time properly to perform the experiment and to record the readings. Do the calculations, draw the graphs and take signature from the instructor.
7. If the experiment is not completed in the stipulated time, the pending work has to be carried out in the leisure hours or extendedhours.
8. Should submit the completed record book according to the deadlines set up by theinstructor.
9. For practical subjects there shall be a continuous evaluation during the semester for 25 sessional marks and 50 end examinationmarks.
10. Out of 25 internal marks, 15 marks shall be awarded for day-to-day work and 10 marks to be awarded by conducting an internal laboratorytest.

WEEK -1
MATRIX OPERATIONS

OBJECTIVE : Write a Python program to find out when given an array of size **N**, the task is to partition the given array into two subsets such that the average of all the elements in both subsets is equal. If no such partition exists print -1. Otherwise, print the partitions. If multiple solutions exist, print the solution where the length of the first subset is minimum. If there is still a tie then print the partitions where the first subset is lexicographically smallest.

RESOURCE : Python 3.73

PROCEDURE:
a. Create : Open a new file in Python shell, write a program and save the program with .py extension.
b. Execute : Go to Run -> Run module (F5)

SOURCE CODE:

```
# Function to print the equal sum sets of the array.
defprintSets(set1, set2) :

    # Print set 1.
    fori inrange(0, len(set1)) :
        print("{} ".format(set1[i]), end ="");
    print("")

    # Print set 2.
    fori inrange(0, len(set2)) :
        print("{} ".format(set2[i]), end ="");
    print("")

# Utility function to find the sets of the
# array which have equal sum.
deffindSets(arr, n, set1, set2, sum1, sum2, pos) :


    # If entire array is traversed, compare both
    # the sums.
    if(pos ==n) :

        # If sums are equal print both sets and
        # return true to show sets are found.
        if(sum1 ==sum2) :
printSets(set1, set2)
            returnTrue

        # If sums are not equal then return
        # sets are not found.
        else:
            returnFalse

    # Add current element to set 1.
    set1.append(arr[pos])

    # Recursive call after adding current
    # element to set 1.
    res =findSets(arr, n, set1, set2,
               sum1 +arr[pos], sum2, pos +1)
```

```
    # If this inclusion results in equal sum
    # sets partition then return true to show
    # desired sets are found.
    if(res) :
        returnres

    # If not then backtrack by removing current
# element from set1 and include it in set 2.
    set1.pop()
    set2.append(arr[pos])

    # Recursive call after including current
    # element to set 2.
    returnfindSets(arr, n, set1, set2, sum1,
                   sum2 +arr[pos], pos +1)

# Return true if array arr can be partitioned
# into two equal sum sets or not.
defisPartitionPoss(arr, n) :

    # Calculate sum of elements in array.
    sum=0

    fori inrange(0, n):
        sum+=arr[i]

    # If sum is odd then array cannot be
    # partitioned.
    if(sum%2!=0) :
 returnFalse

    # Declare vectors to store both the sets.
    set1 =[]
    set2 =[]

    # Find both the sets.
    returnfindSets(arr, n, set1, set2, 0, 0, 0)

# Driver code
arr =[5, 5, 1, 11]
n =len(arr)
if(isPartitionPoss(arr, n) ==False) :
    print("-1")
```

```
Input :arr = {5, 5, 1, 11}
Output : Set 1 = {5, 5, 1},
         Set 2 = {11}
Sum of both the sets is 11 and equal.

Input :arr = {1, 5, 3}
Output : -1
No partitioning results in equal sum sets.
```

VIVA QUESTIONS:

1. What are the different assignment operator used in python?
2. What are the rules to write an identifier?
3. What is a keyword in python?
4. Which data types are supported in python?
5. What are the Boolean values used in python?

WEEK -2
MATRIX OPERATIONS

OBJECTIVE : Write a Python program to find out when given an array of positive elements, you have to flip the sign of some of its elements such that the resultant sum of the elements of array should be minimum non-negative (as close to zero as possible). Return the minimum no. of elements whose sign needs to be flipped such that the resultant sum is minimum non-negative. Note that the sum of all the array elements will not exceed 10^4.

RESOURCE : Python 3.73

PROCEDURE:

a. Create : Open a new file in Python shell, write a program and save the program with .py extension.

b. Execute : Go to Run -> Run module (F5)

SOURCE CODE:

```
# Function to return the minimum number of elements
# whose sign must be flipped to get the positive
# sum of array elements as close to 0 as possible
defsolve(A, n):


    dp=[[0fori inrange(2000)] fori inrange(2000)]

    # boolean variable used for toggling between maps
    flag =1

    # Calculate the sum of all elements of the array
    sum=0
    fori inrange(n):
        sum+=A[i]

    # Initializing first map(dp[0]) with INT_MAX because
    # fori=0, there are no elements in the array to flip
    fori inrange(-sum,sum+1):
        dp[0][i] =10**9

    # Base Case
    dp[0][0] =0

    fori inrange(1,n+1):
        forj inrange(-sum,sum+1):
            dp[flag][j] =10**9
            if(j -A[i -1] <=sumandj -A[i -1] >=-sum):
                dp[flag][j] =dp[flag ^ 1][j -A[i -1]]
            if(j +A[i -1] <=sum
                andj +A[i -1] >=-sum
anddp[flag ^ 1][j +A[i -1]] !=10**9):
                dp[flag][j] =min(dp[flag][j],
                                 dp[flag ^ 1][j +A[i -1]] +1)

        # For toggling
        flag =flag ^ 1

    # Required sum is minimum non-negative
    # So, we iterate from i=0 to sum and find
    # the first i where dp[flag ^ 1][i] != INT_MAX
    fori inrange(sum+1):
        if(dp[flag ^ 1][i] !=10**9):
            returndp[flag ^ 1][i]
```

```
    # In worst case we will flip max n-1 elements
    returnn -1

# Driver code
arr=[10, 22, 9, 33, 21, 50, 41, 60]
n =len(arr)
print(solve(arr, n))
```

Input: arr[] = [14, 10, 4]
Output: 1
Here, we will flip the sign of 14 and the resultant sum will be 0.
Note that flipping the signs of 10 and 4 also gives
the resultant sum 0 but the count of flipped elements is not minimum.

VIVA QUESTIONS:

1. What are the binary operators used in python?
2. What are the different comparison operators used in python?
3. What are the different string literals?
4. What is string concatenation?
5. What is string replication?

WEEK -3

MINIMUM COST PATH

OBJECTIVE : Write a Python program to find out when given a two dimensional grid, each cell of which contains integer cost which represents a cost to traverse through that cell. The task is to find the maximum cost path from the bottom-left corner to the top-right corner.

RESOURCE : Python 3.73

PROCEDURE:

a. Create : Open a new file in Python shell, write a program and save the program with .py extension.

b. Execute : Go to Run -> Run module (F5)

SOURCE CODE:

```
#define ROW 5
#define COL 5

// structure for information of each cell
structcell
{
    intx, y;
    intdistance;
    cell(intx, inty, intdistance) :
        x(x), y(y), distance(distance) {}
};

// Utility method for comparing two cells
booloperator<(constcell& a, constcell& b)
{
    if(a.distance == b.distance)
    {
        if(a.x != b.x)
            return(a.x<b.x);
        else
            return(a.y<b.y);
    }
 return(a.distance<b.distance);
}

// Utility method to check whether a point is
// inside the grid or not
boolisInsideGrid(inti, intj)
{
    return(i>= 0 &&i< COL && j >= 0 && j < ROW);
}
```

VIVA QUESTIONS:

1. What is the print function?
2. What is the input function?
3. What is an expression made up of? What do all expressions do?
4. What three functions can be used to get the integer, floating-point number, or string version of a value?
5. Name three data types.

WEEK -4

FINDING MAXIMUM IN AN INTEGER ARRAY

OBJECTIVE : Write a Python program to find out when given an array of non-negative integers **arr[]**, the task is to find a pair **(n, r)** such that ${}^{n}P_{r}$ is maximum possible and **r ≤ n**.

RESOURCE : Python 3.73

PROCEDURE:

a. Create : Open a new file in Python shell, write a program and save the program with .py extension.

b. Execute : Go to Run -> Run module (F5)

SOURCE CODE:

```
# Function to print the pair (n, r)
# such that nPr is maximum possible
deffindPair(arr, n):

    # There should be atleast 2 elements
    if(n < 2):
        print("-1")
        return

    i =0
    first =-1
    second =-1

    # Findex the largest 2 elements
    fori inrange(n):
        if(arr[i] > first):
            second =first
            first =arr[i]
        elif(arr[i] > second):
            second =arr[i]

    print("n =", first, "and r =", second)

# Driver code
arr =[0, 2, 3, 4, 1, 6, 8, 9]
n =len(arr)

findPair(arr, n)
```

Input: arr[] = {5, 2, 3, 4, 1}

Output: n = 5 and r = 4

${}^{5}P_{4}$ = 5! / (5 – 4)! = 120 which is maximum possible.

Input: arr[] = {0, 2, 3, 4, 1, 6, 8, 9}

Output: n = 9 and r = 8

VIVA QUESTIONS:

1. How an if statement is executed in python?
2. How an else statement is executed in python?
3. What is the use of elif statement?
4. What is the use of while loop?
5. What is the keyword used to end the while loop?

WEEK -5
ARRAY SORTING

OBJECTIVE : Write a Python program to find out when given an array of non-negative integers **arr[]**, the task is to find a pair **(n, r)** such that ${}^{n}P_{r}$ is maximum possible and **r ≤ n**.

RESOURCE : Python 3.73

PROCEDURE:

a. Create : Open a new file in Python shell, write a program and save the program with .py extension.

b. Execute : Go to Run -> Run module (F5)

SOURCE CODE:

```
# Function to return the minimum number
# of given operations required
# to sort the array
defgetMinimumOps(ar):

    # Number of elements in the array
    n =len(ar)

    # Smallest element in the array
    small =min(ar)

    # Largest element in the array
    large =max(ar)

    """
        dp(i, j) represents the minimum number
        of operations needed to make the
        array[0 .. i] sorted in non-decreasing
        order given that ith element is j
    """
    dp =[[ 0fori inrange(large +1)]
              fori inrange(n)]

# Fill the dp[]][ array for base cases
    forj inrange(small, large +1):
        dp[0][j] =abs(ar[0] -j)
    """
    /*
        Using results for the first (i - 1)
        elements, calculate the result
        for the ith element
    */
    """
    fori inrange(1, n):
        minimum =10**9
        forj inrange(small, large +1):

        # """
        #     /*
        #     If the ith element is j then we can have
        #     any value from small to j for the i-1 th
        #     element
        #     We choose the one that requires the
        #     minimum operations
        # """
            minimum =min(minimum, dp[i -1][j])
            dp[i][j] =minimum +abs(ar[i] -j)
"""
```

```
    /*
        If we made the (n - 1)th element equal to j
        we required dp(n-1, j) operations
        We choose the minimum among all possible
        dp(n-1, j) where j goes from small to large
    */
    """
    ans =10**9
    forj inrange(small, large +1):
        ans =min(ans, dp[n -1][j])

    returnans

# Driver code
ar =[1, 2, 1, 4, 3]

print(getMinimumOps(ar))
```

Input: arr[] = {1, 2, 1, 4, 3}
Output: 2
Add 1 to the 3rd element(1) and subtract 1 from
the 4th element(4) to get {1, 2, 2, 3, 3}
Input: arr[] = {1, 2, 2, 100}
Output: 0
Given array is already sorted.

VIVA QUESTIONS:

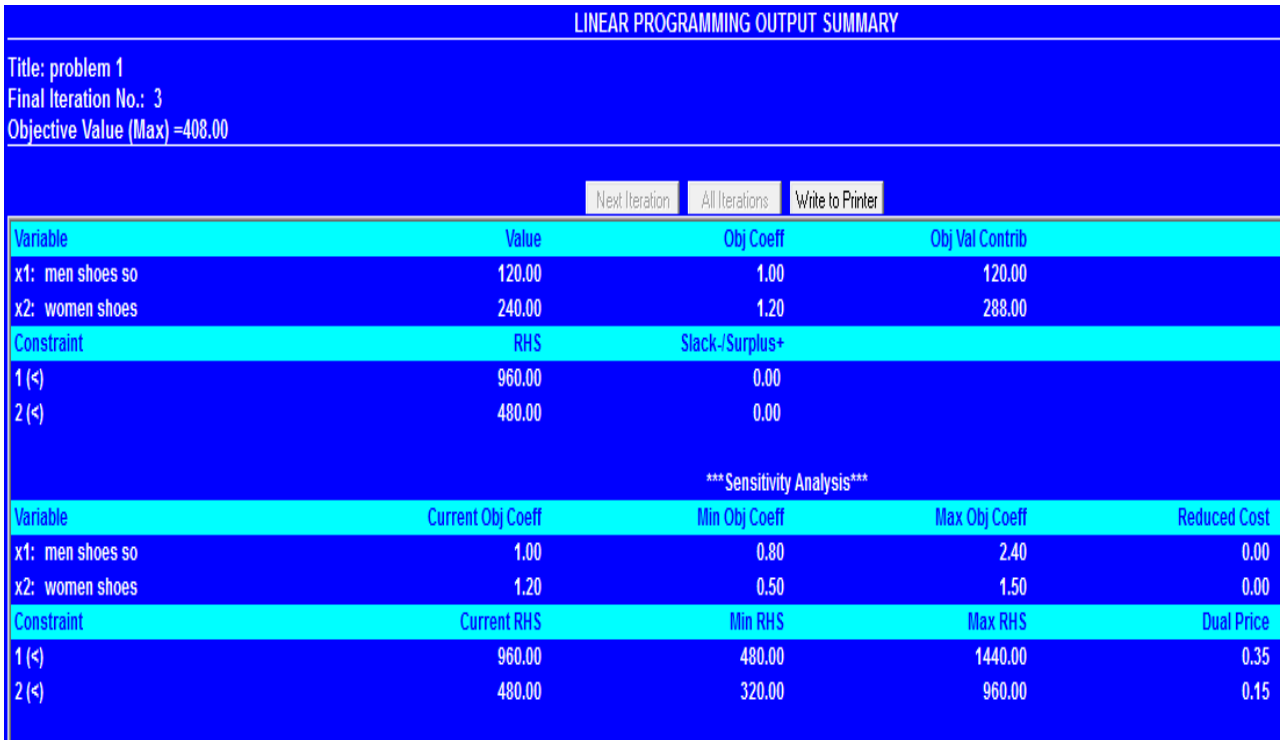
1. What is the difference between break and continue?
2. What keys can you press if your program is stuck in an infinite loop?
3. Write code that prints Hello if 1 is stored in spam, prints Howdy if 2 is stored in spam, and prints Greetings! if anything else is stored in spam.
4. What are the six comparison operators?
5. Explain what a condition is and where you would use one.

WEEK -6
LINEAR PROGRAMMING PROBLEM

OBJECTIVE :A store sells men's and women's tennis shoes. It makes a profit of $1 per pair of men's shoes and $1.20 per pair of women's shoes. It takes two minutes of a salesperson's time and two minutes of a cashier's time to sell a pair of men's shoes. It takes three minutes of a salesperson's time and one minute of a cashier's time per pair of women's shoes. The store is open eight hours per day, during which time there are two salespersons and one cashier on duty. How many pairs of shoes of each type should the store sell in order to maximize profit each day?

RESOURCE : TORA

PROCEDURE:
Step 1: click to exe file of TORA, to open the software.
Step 2 : click linear programming problem.
Step 3: After click to linear programming a window is open (as shown). Click to the "Go to the input screen".
Step 4: After that write Problem Title, No. of Variable and No. of Constraint according to the given problem.
Step 5: A new window is open. Fill all column according to the problem. Such that fill column of Maximize or Minimize and then constraint. And click to the Solve Now.



LINEAR PROGRAMMING OUTPUT SUMMARY

Title: problem 1
Final Iteration No.: 3
Objective Value (Max) =408.00

Next Iteration | All Iterations | Write to Printer

Variable	Value	Obj Coeff	Obj Val Contrib
x1: men shoes so	120.00	1.00	120.00
x2: women shoes	240.00	1.20	288.00
Constraint	**RHS**	**Slack-/Surplus+**	
1 (<)	960.00	0.00	
2 (<)	480.00	0.00	

Sensitivity Analysis

Variable	Current Obj Coeff	Min Obj Coeff	Max Obj Coeff	Reduced Cost
x1: men shoes so	1.00	0.80	2.40	0.00
x2: women shoes	1.20	0.50	1.50	0.00
Constraint	**Current RHS**	**Min RHS**	**Max RHS**	**Dual Price**
1 (<)	960.00	480.00	1440.00	0.35
2 (<)	480.00	320.00	960.00	0.15

VIVA QUESTIONS:
1. What are the non negativity constraints?
2. What are the different constraints?
3. Define objective function?
4. What is an optimal solution?
5. What is basic feasible and non feasible solutions?

WEEK -7
QUEUING PROBLEM

OBJECTIVE :A super market has two girls ringing up sales at the counters. If the service time for each customer is exponential with mean 4 minutes, and if people arrive 3 in a poison fashion at the 10/hour.

RESOURCE : TORA

PROCEDURE:
Step 1: click to exe file of TORA, to open the software.
Step 2 : click queuing programming problem.
Step 3: After click to linear programming a window is open (as shown). Click to the "Go to the input screen".
Step 4: After that write Problem Title, No. of Variable and No. of Constraint according to the given problem.
Step 5: A new window is open. Fill all column according to the problem. Such that fill column of Maximize or Minimize and then constraint. And click to the Solve Now.

LINEAR PROGRAMMING OUTPUT SUMMARY

Title: problem2
Final Iteration No.: 3
Objective Value (Max) =39.53

Next Iteration | All Iterations | Write to Printer

Variable	Value	Obj Coeff	Obj Val Contrib
x1: no of produc	14.12	2.00	28.24
x2: no of produc	3.76	3.00	11.29
Constraint	**RHS**	**Slack-/Surplus+**	
1 (<)	160.00	0.00	
2 (<)	160.00	0.00	
3 (<)	160.00	9.41-	

Sensitivity Analysis

Variable	Current Obj Coeff	Min Obj Coeff	Max Obj Coeff	Reduced Cost
x1: no of produc	2.00	0.90	6.00	0.00
x2: no of produc	3.00	1.00	6.67	0.00
Constraint	**Current RHS**	**Min RHS**	**Max RHS**	**Dual Price**
1 (<)	160.00	40.00	176.00	0.13
2 (<)	160.00	96.00	186.67	0.12
3 (<)	160.00	150.59	infinity	0.00

VIVA QUESTIONS:

1. What are the applications of queuing problems?
2. What are the different types of queuing problems?
3. Define queue length?
4. Define waiting time?
5. What is FCFS?

.WEEK -8
SEQUENCING PROBLEM

OBJECTIVE : We have five jobs each of which must go through two machines in the order BA, processing times are given in the table below

Job No.	1	2	3	4	5
Machine A	10	2	18	6	20
Machine B	4	12	14	16	8

Determine a sequence for the five jobs that will minimize the total elapsed time. Also compute idle times for each of the machine

RESOURCE : TORA

PROCEDURE:
Step 1: click to exe file of TORA, to open the software.
Step 2 : click queuing programming problem.
Step 3: After click to linear programming a window is open (as shown). Click to
the "Go to the input screen".
Step 4: After that write Problem Title, No. of Variable and No. of Constraint according to the given problem.
Step 5: A new window is open. Fill all column according to the problem. Such that fill column of Maximize or Minimize and then constraint. And click to the Solve Now.

LINEAR PROGRAMMING OUTPUT SUMMARY

Title: problem 3
Final Iteration No.: 10
Objective Value (Min) =0.41

Next Iteration | All Iterations | Write to Printer

Variable	Value	Obj Coeff	Obj Val Contrib
x1: Ing1 in mix	0.00	0.55	0.00
x2: Ing2 in mix	0.05	0.42	0.02
x3: Ing3 in mix	1.08	0.36	0.39
Constraint	**RHS**	**Slack-/Surplus+**	
1 (>)	35.00	0.00	
2 (>)	8.00	0.00	
3 (<)	10.00	2.00-	
4 (>)	180.00	32.10+	
5 (>)	9.00	2.05+	

Sensitivity Analysis

Variable	Current Obj Coeff	Min Obj Coeff	Max Obj Coeff	Reduced Cost
x1: Ing1 in mix	0.55	0.50	infinity	-0.05
x2: Ing2 in mix	0.42	0.12	0.49	0.00
x3: Ing3 in mix	0.36	0.29	0.46	0.00
Constraint	**Current RHS**	**Min RHS**	**Max RHS**	**Dual Price**
1 (>)	35.00	26.40	36.57	0.00
2 (>)	8.00	7.66	10.00	0.04
3 (<)	10.00	8.00	infinity	0.00
4 (>)	180.00	-infinity	212.10	0.00
5 (>)	9.00	-infinity	11.05	0.00

VIVA QUESTIONS:

1. What are the applications of sequencing problems?
2. How many types of sequencing models exist?
3. What is total elapsed time?
4. What is Johnsons rule?
5. What is no passing rule?

.WEEK -9
GAME THEORY

OBJECTIVE : Using the dominance property obtain the optimal strategy for both the players and determine the value of game. The payoff matrix for player A is given

Player-A	Player-B					
		I	II	III	IV	V
	I	2	4	3	8	4
	II	5	6	8	7	8
	III	6	7	9	8	7
	IV	4	2	8	4	3

RESOURCE : TORA

PROCEDURE:
Step 1: click to exe file of TORA, to open the software.
Step 2 : click queuing programming problem.
Step 3: After click to linear programming a window is open (as shown). Click to the "Go to the input screen".
Step 4: After that write Problem Title, No. of Variable and No. of Constraint according to the given problem.
Step 5: A new window is open. Fill all column according to the problem. Such that fill column of Maximize or Minimize and then constraint. And click to the Solve Now.


```
 Feasible solution found.
 Infeasibilities:                          0.000000
 Total solver iterations:                         0
 Elapsed runtime seconds:                      0.03

 Model Class:                                    LP

 Total variables:                3
 Nonlinear variables:            0
 Integer variables:              0

 Total constraints:              3
 Nonlinear constraints:          0

 Total nonzeros:                 9
 Nonlinear nonzeros:             0


                        Variable           Value
                              X1        0.000000
                              X2        0.000000
                              X3        0.000000

                             Row    Slack or Surplus
                               1        6.000000
                               2        3.000000
                               3        1.000000
```

.WEEK -10
ASSIGNMENT PROBLEM

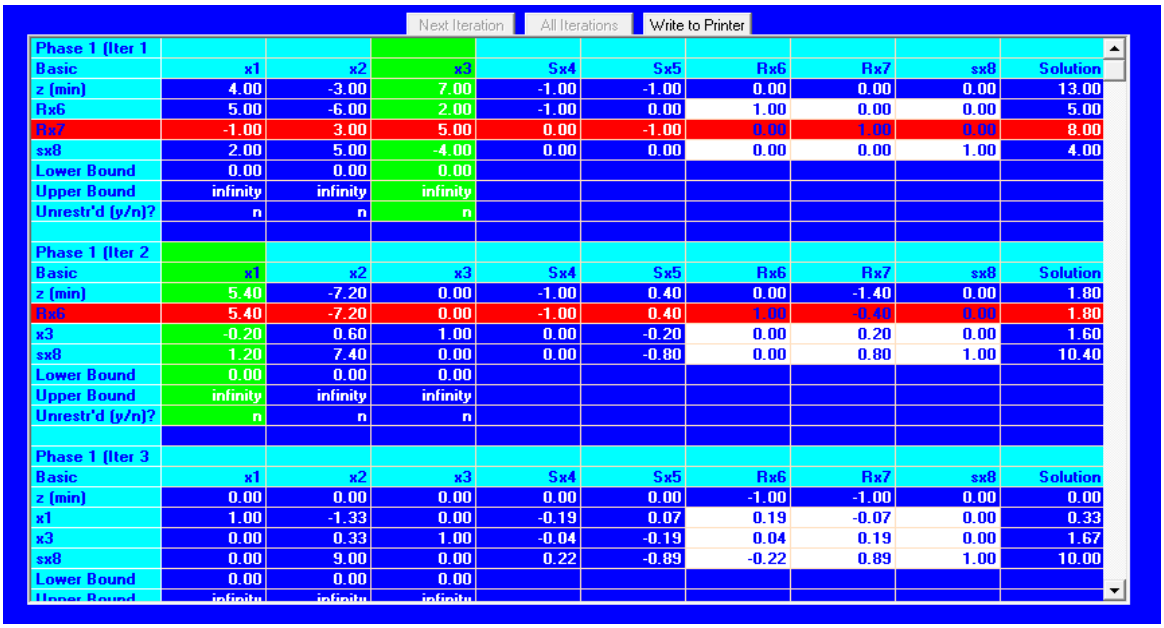
OBJECTIVE :A Company has three plants at locations A,B and C which supply to warehouses located at D,E,F,G and H. monthly plant capacities are 800,500 and 900 respectively. Monthly warehouse requirements are 400, 500,400 and 800 units respectively. Unit transportation cost in rupees is given below.

		Ware houses				
		D	E	F	G	H
Plant	A	5	8	6	6	3
	B	4	7	7	6	5
	C	8	4	6	6	4

Determine an optimum distribution for the company in order to minimize the total transportation cost.

RESOURCE : TORA

PROCEDURE:
Step 1: click to exe file of TORA, to open the software.
Step 2 : click queuing programming problem.
Step 3: After click to linear programming a window is open (as shown). Click to
the "Go to the input screen".
Step 4: After that write Problem Title, No. of Variable and No. of Constraint according to the given problem.
Step 5: A new window is open. Fill all column according to the problem. Such that fill column of Maximize or Minimize and then constraint. And click to the Solve Now.



Phase 1 (Iter 1									
Basic	x1	x2	x3	Sx4	Sx5	Rx6	Rx7	sx8	Solution
z (min)	4.00	-3.00	7.00	-1.00	-1.00	0.00	0.00	0.00	13.00
Rx6	5.00	-6.00	2.00	-1.00	0.00	1.00	0.00	0.00	5.00
Rx7	-1.00	3.00	5.00	0.00	-1.00	0.00	1.00	0.00	8.00
sx8	2.00	5.00	-4.00	0.00	0.00	0.00	0.00	1.00	4.00
Lower Bound	0.00	0.00	0.00						
Upper Bound	infinity	infinity	infinity						
Unrestr'd (y/n)?	n	n	n						
Phase 1 (Iter 2									
Basic	x1	x2	x3	Sx4	Sx5	Rx6	Rx7	sx8	Solution
z (min)	5.40	-7.20	0.00	-1.00	0.40	0.00	-1.40	0.00	1.80
Rx6	5.40	-7.20	0.00	-1.00	0.40	1.00	-0.40	0.00	1.80
x3	-0.20	0.60	1.00	0.00	-0.20	0.00	0.20	0.00	1.60
sx8	1.20	7.40	0.00	0.00	-0.80	0.00	0.80	1.00	10.40
Lower Bound	0.00	0.00	0.00						
Upper Bound	infinity	infinity	infinity						
Unrestr'd (y/n)?	n	n	n						
Phase 1 (Iter 3									
Basic	x1	x2	x3	Sx4	Sx5	Rx6	Rx7	sx8	Solution
z (min)	0.00	0.00	0.00	0.00	0.00	-1.00	-1.00	0.00	0.00
x1	1.00	-1.33	0.00	-0.19	0.07	0.19	-0.07	0.00	0.33
x3	0.00	0.33	1.00	-0.04	-0.19	0.04	0.19	0.00	1.67
sx8	0.00	9.00	0.00	0.22	-0.89	-0.22	0.89	1.00	10.00
Lower Bound	0.00	0.00	0.00						
Upper Bound	infinity	infinity	infinity						

VIVA QUESTIONS:

1. What is an assignment problem?
2. What is Hungarian method?
3. How to solve a maximization assignment problem?
4. How to confirm the optimal solution in assignment problem?
5. What are the applications of assignment problem?

.WEEK -11
DYNAMIC PROGRAMMING PROBLEM

OBJECTIVE : Given an array **arr[]** of **N** integers, the task is to sort the array in non-decreasing order by performing the minimum number of operations. In a single operation, an element of the array can either be incremented or decremented by **1**. Print the minimum number of operations required.

RESOURCE : Python 3.73

PROCEDURE:

a. Create : Open a new file in Python shell, write a program and save the program with .py extension.

b. Execute : Go to Run -> Run module (F5)

SOURCE CODE:

```
# Function to return the minimum number
# of given operations required
# to sort the array
defgetMinimumOps(ar):

    # Number of elements in the array
    n =len(ar)

    # Smallest element in the array
    small =min(ar)

    # Largest element in the array
    large =max(ar)

    """
        dp(i, j) represents the minimum number
        of operations needed to make the
        array[0 .. i] sorted in non-decreasing
        order given that ith element is j
    """
    dp =[[ 0fori inrange(large +1)]
              fori inrange(n)]

    # Fill the dp[]][ array for base cases
    forj inrange(small, large +1):
        dp[0][j] =abs(ar[0] -j)
    """
    /*
        Using results for the first (i - 1)
        elements, calculate the result
        for the ith element
    */
    """
    fori inrange(1, n):
        minimum =10**9
        forj inrange(small, large +1):

            # """
            #     /*
            #     If the ith element is j then we can have
            #     any value from small to j for the i-1 th
            #     element
            #     We choose the one that requires the
            #     minimum operations
            # """
                minimum =min(minimum, dp[i -1][j])
```

```
            dp[i][j] =minimum +abs(ar[i] -j)
    """
    /*
        If we made the (n - 1)th element equal to j
        we required dp(n-1, j) operations
        We choose the minimum among all possible
        dp(n-1, j) where j goes from small to large
    */
    """
    ans =10**9
    forj inrange(small, large +1):
        ans =min(ans, dp[n -1][j])
 returnans

# Driver code
ar =[1, 2, 1, 4, 3]

print(getMinimumOps(ar))
```

Input: arr[] = {1, 2, 1, 4, 3}
Output: 2
Add 1 to the 3rd element(1) and subtract 1 from the 4th element(4) to get {1, 2, 2, 3, 3}

VIVA QUESTIONS:

1. What is the use of dynamic programming problem?
2. What are the applications of operations research?
3. What are the characteristics of dynamic programming problem?
4. What are the flow control statements?
5. What is the difference between interactive shell and editor file?

WEEK -12
INVENTORY PROBLEM

OBJECTIVE :A dealer supplies you the following information with regards to an product that he deals in annual demand =10,000 units, ordering cost Rs.10/order, Price Rs.20/unit. Inventory carrying cost is 20% of the value of inventory per year. The dealer is considering the possibility of allowing some back orders to occurs. He has estimated that the annual cost of back orderingwill be 25% of the value of inventory.

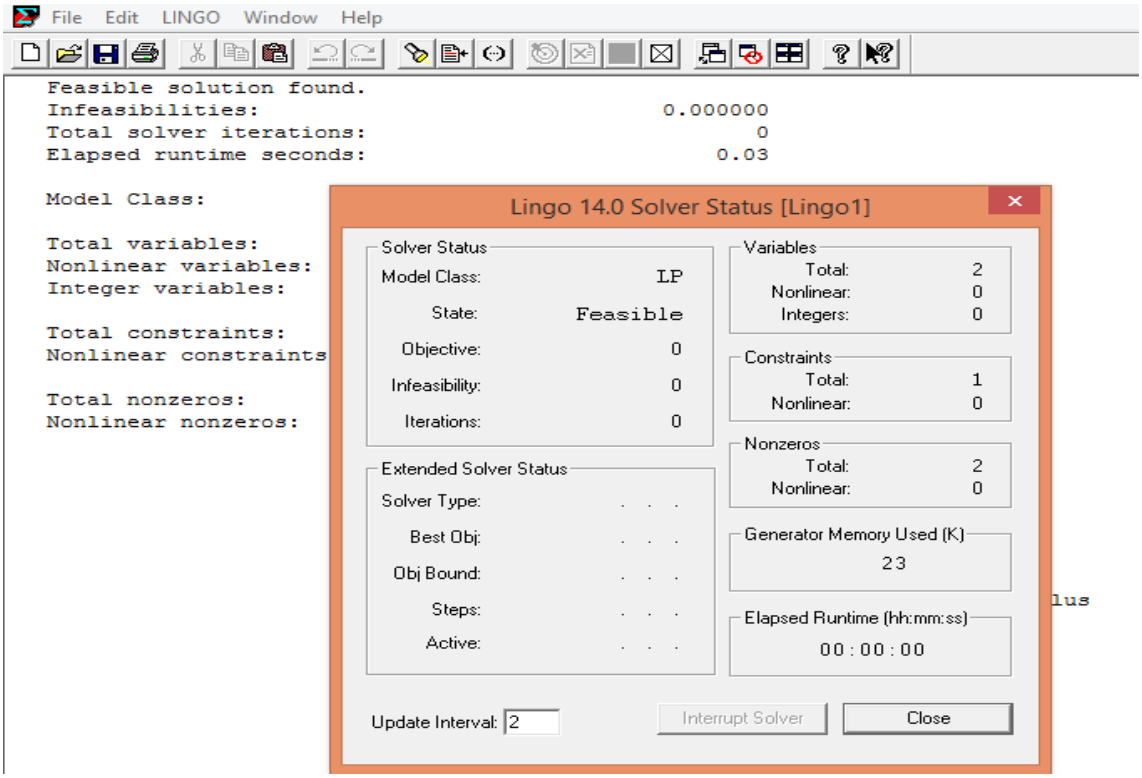
a. What should be the optimum no of units he should buy in 1lot?
b. What qty of the product should be allowed to be backordered
c. What would be the max qty of inventory at any time of year

Would you recommend to allow backordering? If so what would be the annual cost saving by adopting the policy of backordering..

RESOURCE :LINGO

PROCEDURE:

Step 1: click to exe file of TORA, to open the software.
Step 2 : click queuing programming problem.
Step 3: After click to linear programming a window is open (as shown). Click to the "Go to the input screen".
Step 4: After that write Problem Title, No. of Variable and No. of Constraint according to the given problem.
Step 5: A new window is open. Fill all column according to the problem. Such that fill column of Maximize or Minimize and then constraint. And click to the Solve Now.



VIVA QUESTIONS:

1. What is the use of inventory?
2. Define ordering cost?
3. Define inventory carrying cost?