Pune Vidyarthi Griha's

COLLEGE OF ENGINEERING, NASIK

SE COMPUTER ENGINEERING DEPT.

LAB MANUAL

DATA STRUCTURE LABORATORY

Prepared by

PROF. ANAND GHARU

2020 - 21

Pune Vidyarthi Griha's

COLLEGE OF ENGINEERING, NASHIK-4

DEPARTMENT OF COMPUTER ENGINEERING

Lab Manual

Second Year Engineering
Semester-III
DATA STRUCTURES LAB
Subject Code: 210246

Class: Second Year

Prepared By:

Prof. Anand N. Gharu

(Assistant Professor, M.tech Computer Engineering)

Academic year 2020-21

DATA STRUCTURES LAB
(210246)

Teaching Scheme	Credit	Examination Scheme
PR: 04 Hours/Week	02	TW: 25 Marks PR: 50 Marks

Guidelines for Instructor's Manual

The instructor‘s manual is to be developed as a hands-on resource and reference. The instructor's manual need to include prologue (about University/program/ institute/ department/foreword/ preface etc), University syllabus, conduction & Assessment guidelines, topics under consideration-concept, objectives, outcomes, set of typical applications/practicals/ guidelines, and references.

Guidelines for Student Journal

The laboratory practicals are to be submitted by student in the form of journal. Journal consists of prologue, Certificate, table of contents, and handwritten write-up of each practical (Title, Objectives, Problem Statement, Outcomes, software & Hardware requirements, Date of Completion, Assessment grade/marks and assessor's sign, Theory- Concept in brief, algorithm, flowchart, test cases, conclusion/analysis. Program codes with sample output of all perform practical’s are to be submitted as softcopy.

As a conscious effort and little contribution towards Green IT and environment awareness, attaching printed papers as part of write-ups and program listing to journal may be avoided. Use of DVD containing students programs maintained by lab In-charge is highly encouraged. For reference one or two journals may be maintained with program prints at Laboratory.

Guidelines for Assessment

Continuous assessment of laboratory work is done based on overall performance and lab practicals performance of student. Each lab practical assessment will assign grade/marks based on parameters with appropriate weightage. Suggested parameters for overall assessment as well as each lab practical assessment include- timely completion, performance, innovation, efficient codes, punctuality and neatness.

Guidelines for Practical Examination

Both internal and external examiners should jointly set problem statements. During practical assessment, the expert evaluator should give the maximum weightage to the satisfactory implementation of the problem statement. The supplementary and relevant questions may be asked at the time of evaluation to test the student‘s for advanced learning, understanding of the fundamentals, effective and efficient implementation. So encouraging efforts, transparent evaluation and fair approach of the evaluator will not create any uncertainty or doubt in the minds of the students. So adhering to these principles will consummate our team efforts to the promising start of the student's academics.

Guidelines for Laboratory Conduction

The instructor is expected to frame the practicals by understanding the prerequisites, technological aspects, utility and recent trends related to the topic. The practical framing policy need to address the average students and inclusive of an element to attract and promote the intelligent students. The instructor may set multiple sets of practicals and distribute among batches of students. It is appreciated if the practicals are based on real world problems/applications. Encourage students for

appropriate use of Hungarian notation, proper indentation and comments. Use of open source software is to be encouraged.

In addition to these, instructor may assign one real life application in the form of a mini- project based on the concepts learned. Instructor may also set one practical or mini-project that is suitable to respective branch beyond the scope of syllabus.

Set of suggested practical list is provided in groups- A, B, C, D, and E. Each student must perform at least 13 practicals as at least 3 from group A, 3 from group B, 2 from group C, 2 from group D and 3 from group E.

Group A and B assignments should be implemented in Python without using built-in methods for major functionality of assignment. Use List data structure of Python as array**. Group C, D and E assignments should be implemented in C++ language.**

Operating System recommended:- 64-bit Open source Linux or its derivative

Programming tools recommended: - Open Source Python,

Programming tool like Jupyter Notebook, Pycharm, Spyder, G++/GCC.

ractical No.	Laboratory Assignments
GROUP - A	
1	A-2 Write a Python program to store marks scored in subject "Fundamental of Data Structure" by N students in the class. Write functions to compute following: a) The average score of class b) Highest score and lowest score of class c) Count of students who were absent for the test d) Display mark with highest frequency
2	A-5 Write a Python program to compute following operations on String: a) To display word with the longest length b) To determines the frequency of occurrence of particular character in the string c) To check given string is palindrome or not d) To display index of first substring e) To count the occurrences of each word in string
3	A-8 Write a Python program that determines the location of a saddle point of matrix if one exists. An m x n matrix is said to have a saddle point if some entry a[i][j] is the smallest value in row i & the largest value in j.
4	A-9 Write a **Python** program to compute following computation on matrix: a)Addition of two matrices b) Subtraction of two matrices c) Multiplication of two matrices d) Transpose of a matrix
GROUP - B	
5	B-14 Write a Python program to store first year percentage of students in array. Write function for sorting array of floating point numbers in ascending order using a)SelectionSort b) Bubble sort and display top five scores.
6	B-15 Write a Python program to store second year percentage of students in array. Write function for sorting array of floating point numbers in ascending order using a) Insertion sort b) Shell Sort and display top five scores
7	B-16 Write a Python program to store first year percentage of students in array. Write function for sorting array of floating point numbers in ascending order using quick sort and display top five scores.
GROUP - C	
8	C-19 Department of Computer Engineering has student's club named 'Pinnacle Club'. Students of second, third and final year of department can be granted membership on request. Similarly one may cancel the membership of club. First node is reserved for president of club and last node is reserved for

	secretary of club. Write C++ program to maintain club member's information using singly linked list. Store student PRN and Name. Write functions to: a) Add and delete the members as well as president or even secretary. b) Compute total number of members of club c) Display members d) Two linked lists exists for two divisions. Concatenate two lists.
9	C-20 The ticket booking system of Cinemax theater has to be implemented using C++ program. There are 10 rows and 7 seats in each row. Doubly circular linked list has to be maintained to keep track of free seats at rows. Assume some random booking to start with. Use array to store pointers (Head pointer) to each row. On demand a) The list of available seats is to be displayed b) The seats are to be booked c) The booking can be cancelled.
10	C-21 Write C++ program for storing appointment schedule for day. Appointments are booked randomly using linked list. Set start and end time and min and max duration for visit slot. Write functions for- A) Display free slots B) Book appointment C) Sort list based on time D) Cancel appointment (check validity, time bounds, availability) E) Sort list based on time using pointer manipulation
GROUP - D	
11	D-26 In any language program mostly syntax error occurs due to unbalancing delimiter such as (),{},[]. Write C++ program using stack to check whether given expression is well parenthesized or not.
12	D-27 Implement C++ program for expression conversion as infix to postfix and its evaluation using stack based on given conditions: 1. Operands and operator, both must be single character. 2. Input Postfix expression must be in a desired format. 3. Only '+', '-', '*' and '/ ' operators are expected.
GROUP - E	
13	E-29 Queues are frequently used in computer programming, and a typical example is the creation of a job queue by an operating system. If the operating system does not use priorities, then the jobs are processed in the order they enter the system. Write C++ program for simulating job queue. Write functions to add job and delete job from queue.
14	E-31 A double-ended queue (deque) is a linear list in which additions and deletions may be made at either end. Obtain a data representation mapping a deque into a one-dimensional array. Write C++ program to simulate deque with functions to add and delete elements from either end of the deque.
15	E-32 Pizza parlor accepting maximum M orders. Orders are served in first come first served basis. Order once placed cannot be cancelled. Write C++ program to simulate the system using circular queue using array.

Practical No: 01(A)

Practical Title: Write a python program to store marks for N students.

Aim: - Write a Python program to store marks scored in subject "Fundamental of Data Structure" by N students in the class. Write functions to compute following:

a) The average score of class

b) Highest score and lowest score of class

c) Count of students who were absent for the test

d) Display mark with highest frequency

Prerequisite:

- Python Programming

Objectives:

- To understand the use functions for N students record.

Input: N number of students.

Outcome: Resulting average, highest and lowest marks operation.

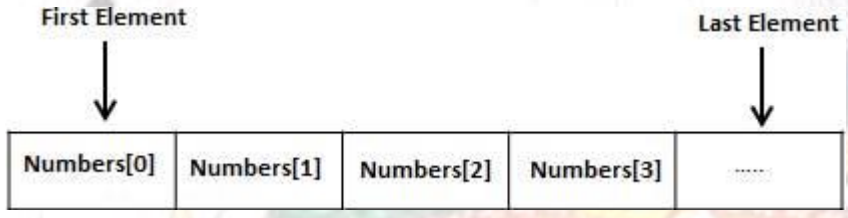
Theory:

ARRAYS

Arrays a kind of data structure that can store a fixed-size sequential collection of elements of the same type. An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type.

Instead of declaring individual variables, such as number0, number1, ..., and number99, you declare one array variable such as numbers and use numbers[0], numbers[1], and ..., numbers[99] to represent individual variables. A specific element in an array is accessed by an index.

All arrays consist of contiguous memory locations. The lowest address corresponds to the first element and the highest address to the last element.



Declaring Arrays

To declare an array in C, a programmer specifies the type of the elements and the number of elements required by an array as follows −

```
type arrayName [ arraySize ];
```

This is called a *single-dimensional* array. The **arraySize** must be an integer constant greater than zero and **type** can be any valid C data type. For example, to declare a 10-element array called **balance** of type double, use this statement −

```
double balance[10];
```

Here *balance* is a variable array which is sufficient to hold up to 10 double numbers.

Initializing Arrays

You can initialize an array in C either one by one or using a single statement as follows −

```
double balance[5] = {1000.0, 2.0, 3.4, 7.0, 50.0};
```

Accessing Array Elements

An element is accessed by indexing the array name. This is done by placing the index of the element within square brackets after the name of the array. For example −

```
double salary = balance[9];
```

Functions Used :

Write algorithm/pseudo code for each function.

a) The average score of class

b) Highest score and lowest score of class

c) Count of students who were absent for the test

d) Display mark with highest frequency

Algorithm:

WRITE ALGO.

Flowchart :

Here, draw flowchart of above algorithm.

Conclusion:

By this way, we can store the marks of N students sucessfully.

A	P	J	Total	Dated Sign
3	4	3	10	

Questions:

1. Basics of python Programming.
2. What are functions?
3. What are the features of python Programming?
4. What are array?

Practical No : 2(A)

Practical Title: Write a python program to perform String operations.

Aim: Write a Python program to compute following operations on String:
a) To display word with the longest length
b) To determines the frequency of occurrence of particular character in the string
c) To check whether given string is palindrome or not
d) To display index of first appearance of the substring
e) To count the occurrences of each word in a given string

Pre-requisite:

- Basics of String operations.

Objectives:

- To understand the use standard library functions for stirng opeations.
- To perform the string operations.

Input: One or Two Strings

Output: Resulting string after performing string operation.

Theory :

String:
String is defined as an array of characters or a pointer to characters.

Null-terminated String:
String is terminated by a special character which is called as null terminator or null parameter (\0). So when you define a string you should be sure to have sufficient space for the null terminator. The null terminator has value 0.

Declaring String:
As in string definition, we have two ways to declare a string. The first way is, we declare an array of characters as follows:

```
char s[] = “string”
 or
char str[20];
```

String operation (explain each operation in detail with example)

a) To display word with the longest length
b) To determines the frequency of occurrence of particular character in the string
c) To check whether given string is palindrome or not
d) To display index of first appearance of the substring
e) To count the occurrences of each word in a given string

Algorithms :
Write algorithms for your program

Flowchart :

Draw flowchart for above algorithm

Conclusion:

By this way, we can perform string operations successfully**.**

A	P	J	Total	Dated Sign
3	4	3	10	

Questions:

1. Write a program to find the length of string.
2. Write a program to display string from backward.
3. Write a program to count number of words in string.
4. Write a program to concatenate one string contents to another.
5. Write a program to compare two strings they are exact equal or not.
6. Write a program to check a string is palindrome or not.
7. WAP to find a substring within a string. If found display its starting position.
8. Write a program to reverse a string.
9. Write a program to convert a string in lowercase.
10. Write a program to convert a string in uppercase.

Practical No : 03(A)

Practical Title: Write program to determine saddle point of matrix.

Aim : Write a Python program that determines the location of a saddle point of matrix if one exists. An m x n matrix is said to have a saddle point if some entry a[i][j] is the smallest value in row i and the largest value in j.

Pre-requisite:

- Knowledge of representing matrix in Python
- Knowledge of different operations that can be performed on matrix

Objectives:

To understand the concept of saddle point in matrix
To implement python program for saddle point

Input :

N x N Matrix to be provided

Outcome:

the location of a saddle point of matrix

Theory:

Concept of saddle point in matrix with example.

Algorithms :
Write algorithm for your program

Flowchart :

Draw flowchart for above algorithm

Conclusion:

By this way, we can determine location of saddle point in matrix successfully.

A	P	J	Total	Dated Sign
3	4	3	10	

Practical No: 4(A)

Practical Title: Perform different operations on Matrix.

Aim : Write a Python program to compute following computation on matrix:
a) Addition of two matrices
b) Subtraction of two matrices
c) Multiplication of two matrices
d) Transpose of a matrix

Pre-requisite:

- Knowledge of representing matrix in Python
- Knowledge of different operations that can be performed on matrix

Objectives:

- Compute the transpose of matrix
- Perform addition , subtraction and multiplication of two matrices.

Input:

- Number of rows and columns of two matrices
- Elements of both the matrices

Outcome:

- Transpose of a matrix
- Result of addition , subtraction and multiplication of both matrices.

Theory:

- **2-dimension array –**

  write theory of 2-D array

- **MatrixOperations (explain each operation in detail with example)**
  - **Concept of matrix**
  - **Addition**
  - **Substraction**
  - **Multiplication**
  - **Transpose of matrix**

Algorithm:

1. Start
2. Input number of rows and columns of first matrix.
3. Input elements of first matrix.
4. Input number of rows and columns of second matrix.
5. Input elements of Second matrix.
6. Function to transpose first matrix i.e. the element at row r column c in the original is placed at row c column r of the transpose.
7. Fuction to add, subtract and multiply two matrices.

Flowchart :

Draw flowchart for above algorithm

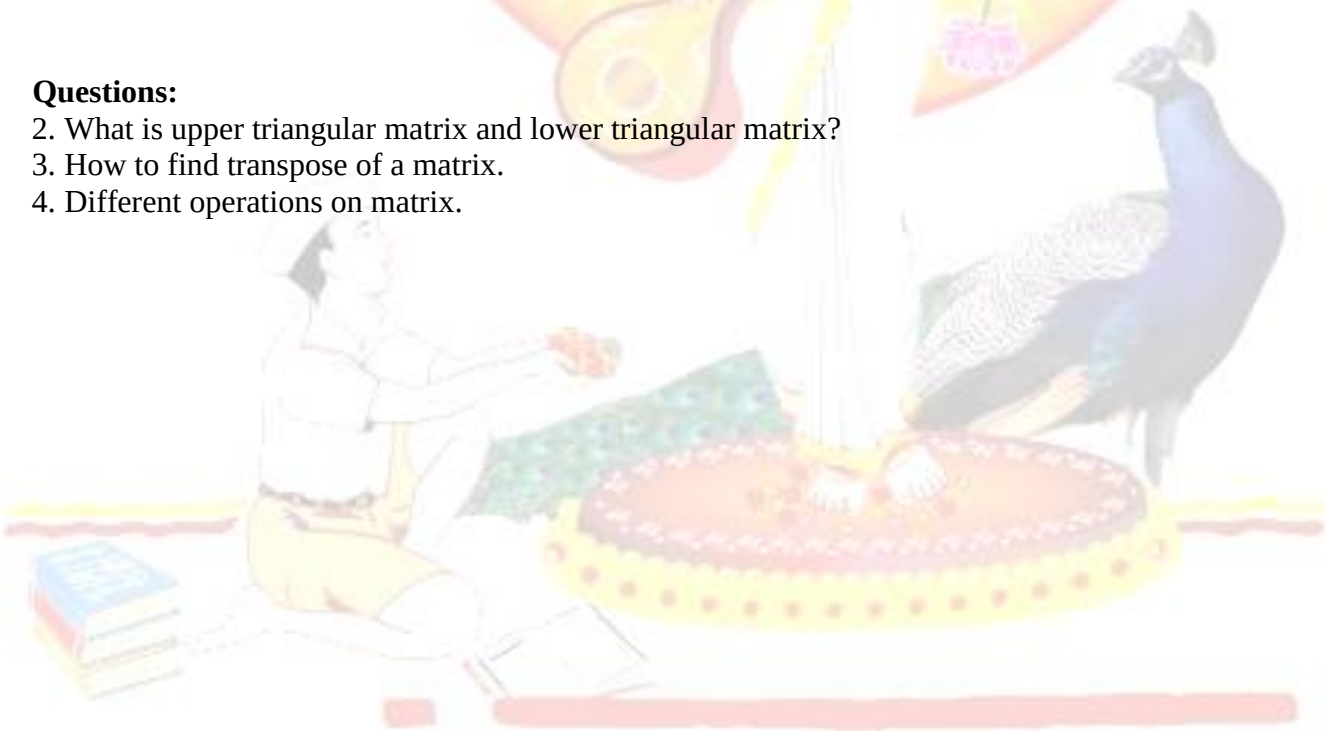
Conclusion:

By this way, we can perform various operations on matrix successfully**.**

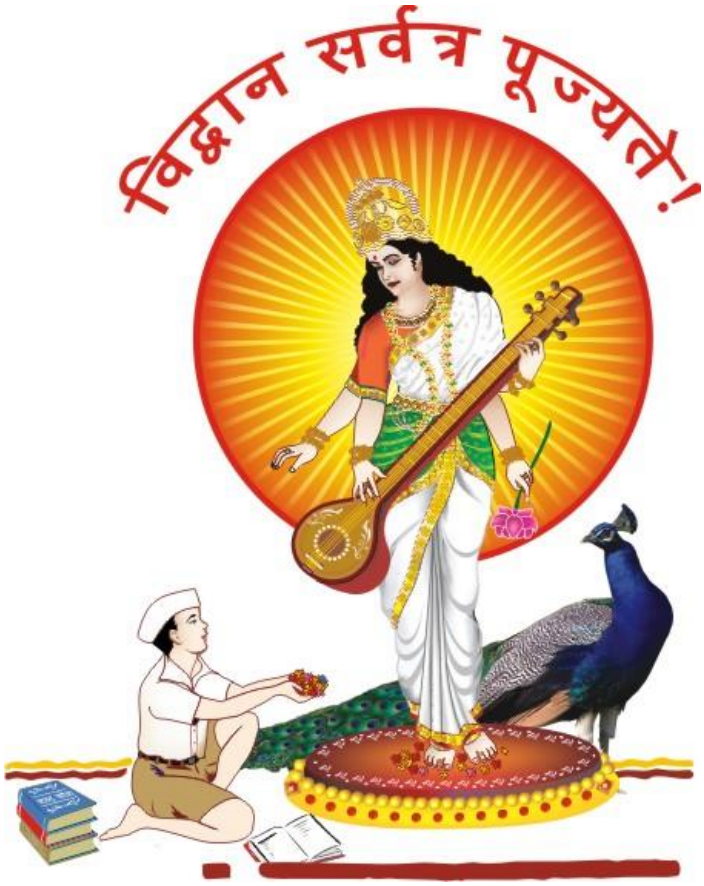
A	P	J	Total	Dated Sign
3	4	3	10	

Questions:

2. What is upper triangular matrix and lower triangular matrix?
3. How to find transpose of a matrix.
4. Different operations on matrix.



GROUP - B



GROUP - B

Practical No:05(B)

Practical Title: Sorting of an array using **selection and bubble sort.**

Aim: Write a Python program to store first year percentage of students in array. Write function for sorting array of floating point numbers in ascending orderusing

a)Selection Sort

b) Bubble sort and display top five scores.of club**.**

Pre-requisite:

Knowledge of sorting techniques

Objective:

To sort array of floating point numbers in ascending order using
a) Selection Sort b) Bubble sort and display top five scores.

Input:

Size of array Elements of array

Theory :

- **Write short theory of sorting with its advantages and disadvantages.**
- **Explain selection and bubble sort with example**

Algorithm:

```
def bubbleSort(alist):
    for passnum in range(len(alist)-1,0,-1):
        for i in range(passnum):
            if alist[i]>alist[i+1]:
                temp = alist[i]
                alist[i] = alist[i+1]
                alist[i+1] = temp

alist = [54,26,93,17,77,31,44,55,20]
bubbleSort(alist)
print(alist)

def selectionSort(alist):
    for fillslot in range(len(alist)-1,0,-1):
        positionOfMax=0
        for location in range(1,fillslot+1):
```

```
        if alist[location]>alist[positionOfMax]:
            positionOfMax = location


    temp = alist[fillslot]
    alist[fillslot] = alist[positionOfMax]
    alist[positionOfMax] = temp


alist = [54,26,93,17,77,31,44,55,20]
selectionSort(alist)
print(alist)
```

Flowchart :

Draw flowchart for above algorithms.

Conclusion:

By this way, we can perform sorting of an array using selection and bubble sort.

A	P	J	Total	Dated Sign
3	4	3	10	

Question Bank:

1. Explain the sorting?
2. What are the different types of sorts in data structures
3. Define the bubble sort?
4. Define the selection sort?
5. How many passes are required in selection sort?
6. What is the time complexity of selecion and bubble sort?

Practical No :6 (B)

Practical Title : Sorting of an array using insertion and shell sort

Aim: Write a Python program to store second year percentage of students in array. Write function for sorting array of floating point numbers in ascending order using

a) Insertion sort

b) Shell Sort and display top five scores

Pre-requisite:

a. Knowledge of sorting techniques

Objective:

b. To sort array of floating point numbers in ascending order using a) Insertion Sort b) Shell sort and display top five scores.

c. Sorted list of elements
d. Top five scores.

Input:

Size of array Elements of array

Theory:

- **Write short theory of sorting.**
- **Explain insertion and shell sort with example**

Algorithm:

```
def selectionSort(alist):
  for fillslot in range(len(alist)-1,0,-1):
     positionOfMax=0
     for location in range(1,fillslot+1):
        if alist[location]>alist[positionOfMax]:
           positionOfMax = location

     temp = alist[fillslot]
     alist[fillslot] = alist[positionOfMax]
     alist[positionOfMax] = temp

alist = [54,26,93,17,77,31,44,55,20]
selectionSort(alist)
print(alist)
```

```
def shellSort(alist):
    sublistcount = len(alist)//2
    while sublistcount > 0:

      for startposition in range(sublistcount):
        gapInsertionSort(alist,startposition,sublistcount)

      print("After increments of size",sublistcount,
                          "The list is",alist)

      sublistcount = sublistcount // 2

def gapInsertionSort(alist,start,gap):
    for i in range(start+gap,len(alist),gap):

        currentvalue = alist[i]
        position = i

        while position>=gap and alist[position-gap]>currentvalue:
            alist[position]=alist[position-gap]
            position = position-gap

        alist[position]=currentvalue

alist = [54,26,93,17,77,31,44,55,20]
shellSort(alist)
print(alist)
```

Flowchart :

Draw flowchart for above algorithms

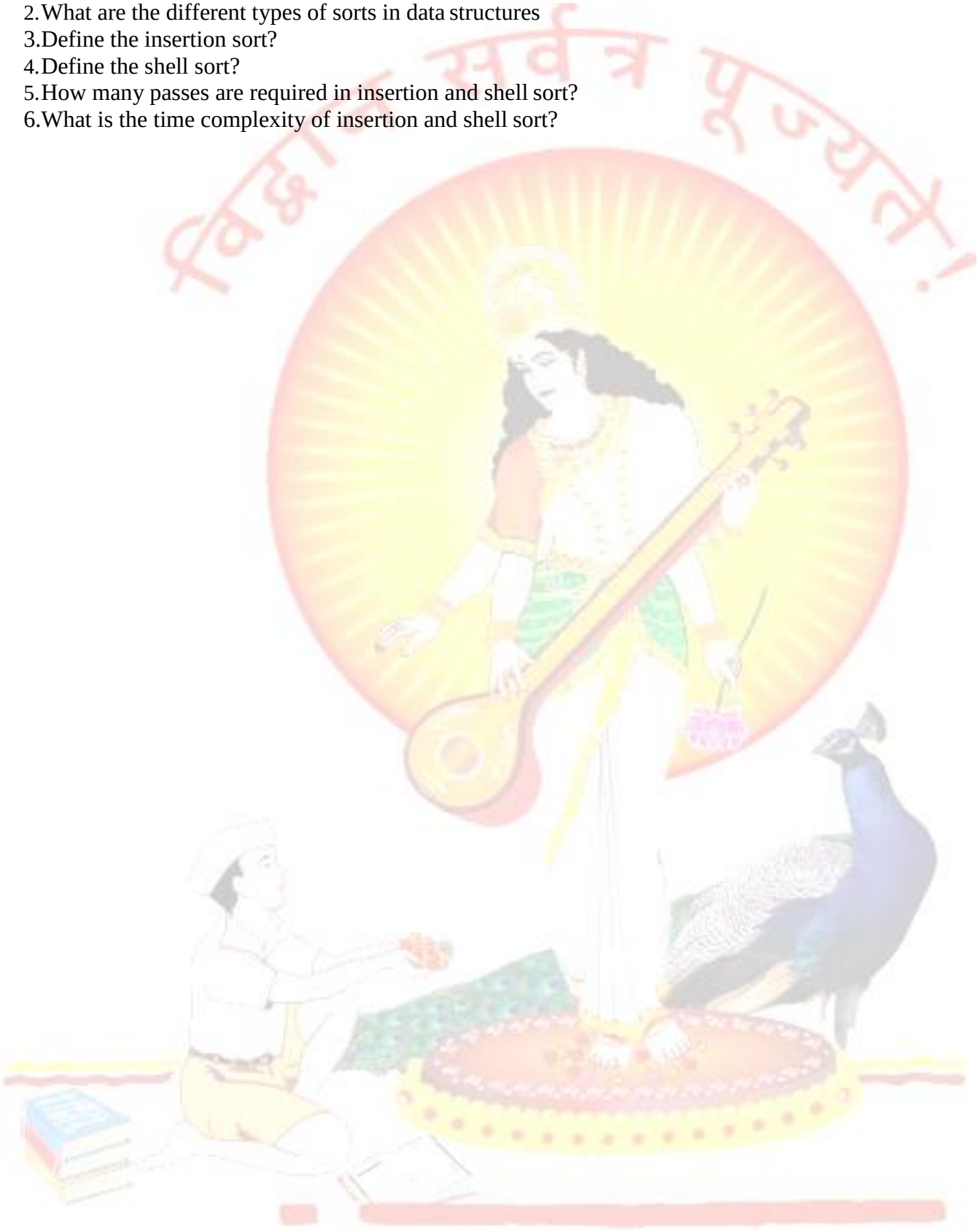
Conclusion:

By this way, we can sort percentage of students in array using insertion sort and shell sort.

A	P	J	Total	Dated Sign
3	4	3	10	

Question Bank:

1. Explain the sorting?
2. What are the different types of sorts in data structures
3. Define the insertion sort?
4. Define the shell sort?
5. How many passes are required in insertion and shell sort?
6. What is the time complexity of insertion and shell sort?



Practical No:07(B)

Practical Title: Sorting array of floating point numbers in ascending order using **quick sort**.

Aim: Write a Python program to store first year percentage of students in array. Write function for sorting array of floating point numbers in ascending order using quick sort and display top five scores.

Pre-requisite:

- Knowledge of sorting techniques

Objective:

- To sort array using quick sort

Input:

Size of array

First year percentage of students.

Outcome:

- To sort array using quick sort
- To display top five scores.

Theory :

- **Explain concept of Quick sort in details.**
- **Advantages and disadvantages**
- **Example of quick sort.**
- **Time complexity.**

Algorithm:

```
def quickSort(alist):
   quickSortHelper(alist,0,len(alist)-1)
def quickSortHelper(alist,first,last):
   if first<last:
      splitpoint = partition(alist,first,last)
      quickSortHelper(alist,first,splitpoint-1)
      quickSortHelper(alist,splitpoint+1,last)
def partition(alist,first,last):
   pivotvalue = alist[first]
   leftmark = first+1
   rightmark = last
   done = False
   while not done:
      while leftmark <= rightmark and alist[leftmark] <= pivotvalue:
```

```
        leftmark = leftmark + 1
    while alist[rightmark] >= pivotvalue and rightmark >= leftmark:
        rightmark = rightmark -1

    if rightmark < leftmark:
        done = True
    else:
        temp = alist[leftmark]
        alist[leftmark] = alist[rightmark]
        alist[rightmark] = temp
  temp = alist[first]
  alist[first] = alist[rightmark]
  alist[rightmark] = temp
  return rightmark
alist = [54,26,93,17,77,31,44,55,20]
quickSort(alist)
print(alist)
```

Flowchart :

Draw flowchart for above algorithms.

Conclusion:

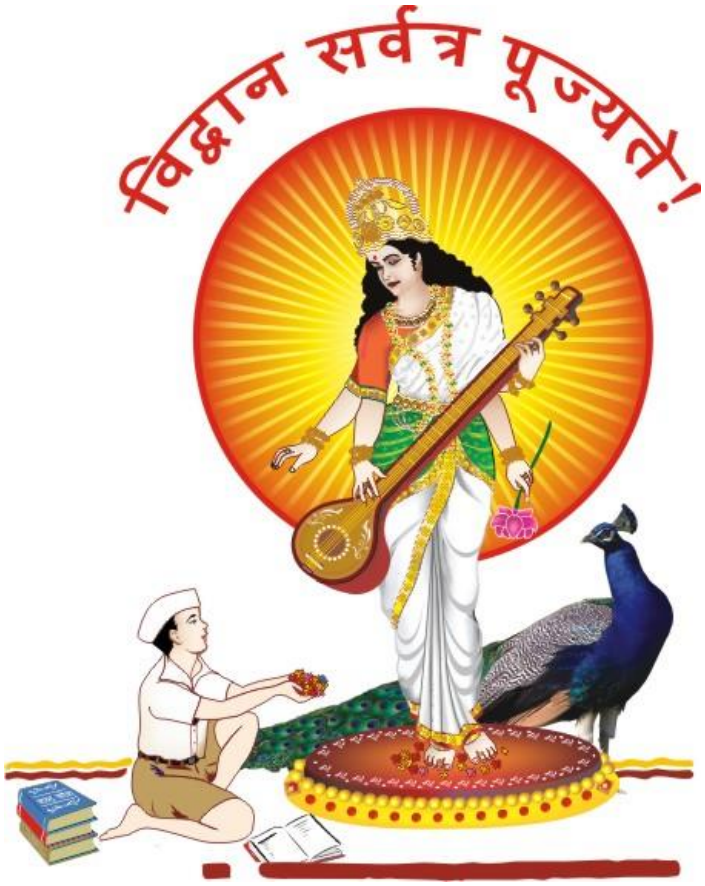
By this way, we can perform sorting array of floating point numbers in ascending order using quick sort.

A	P	J	Total	Dated Sign
3	4	3	10	

Question Bank:

1.Explain the sorting?
2.What are the different types of sorts in data structures?
3.Define the quick sort?
5.How many passes are required in quick sort?
6.What is the time complexity of quick sort?

GROUP - C



GROUP - C

Practical No : 08 (C)

Assignmet Title: Write C++ program to maintain club member‘s information using singly linked list.

Aim: Department of Computer Engineering has student's 1/8 club named 'Pinnacle Club'. Students of Second, third and final year of department can be granted membership on request. Similarly one may cancel the membership of club. First node is reserved for president of club and last node is reserved for secretary of club. Write C++ program to maintain club member‘s information using singly linked list. Store student PRN and Name. Write functions to

a) Add and delete the members as well as president or even secretary.
b)Compute total number of members of club
c)Display members
d)Display list in reverse order using recursion
e) Two linked lists exists for two divisions. Concatenate two lists

Input: Individual details

Output: Maintain information of the Club member's

Objectives:

- To maintain club member's information by performing different operations like add, delete, reverse, concatenate on singly linked list.

Theory:

Linked List : (write linked list definition and sinly linked list theory)

- **Definition :**
- **Types of linked list**
- **Singly linked list (definition, concepts, advantages, disadvantages)**
- **Singly linked list as an ADT (Write pseudo code for each ADT)**

- **Algorithm**

  (Write your own algorithm for your program)

- **Flowchart :**

  (draw your own flowchart for your algorithms)

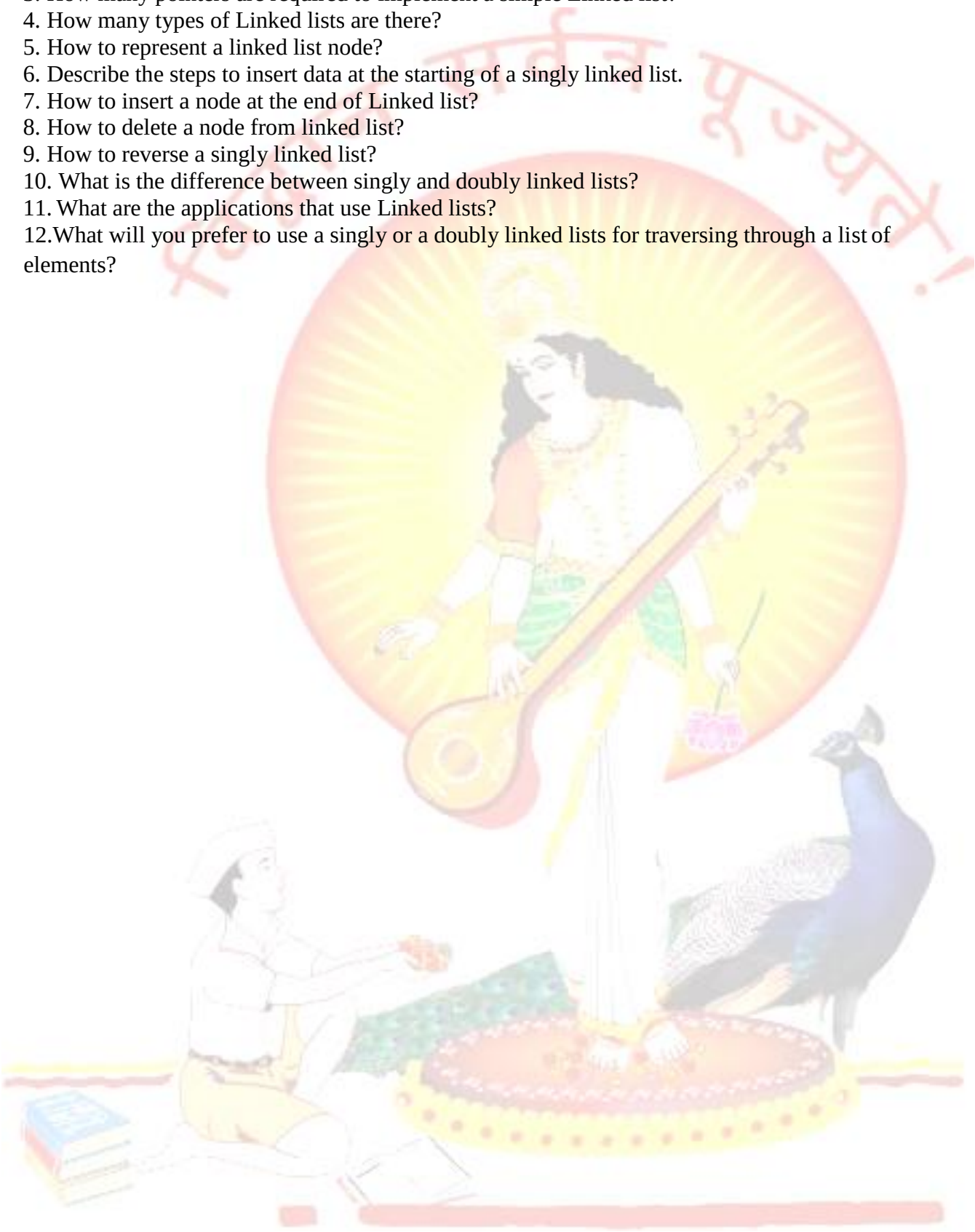
Conclusion:

By this way, we can maintain club member‘s information using singly linked list.

A	P	J	Total	Dated Sign
3	4	3	10	

Questions: (No need to write answers)

1. What is a Linked list?
2. Can you represent a Linked list graphically?
3. How many pointers are required to implement a simple Linked list?
4. How many types of Linked lists are there?
5. How to represent a linked list node?
6. Describe the steps to insert data at the starting of a singly linked list.
7. How to insert a node at the end of Linked list?
8. How to delete a node from linked list?
9. How to reverse a singly linked list?
10. What is the difference between singly and doubly linked lists?
11. What are the applications that use Linked lists?
12. What will you prefer to use a singly or a doubly linked lists for traversing through a list of elements?



Practical No: 09 (C)

Practical Title: The ticket booking system of Cinemax theater has to be implemented using C++ program. There are 10 rows and 7 seats in each row. Doubly circular linked list has to be maintained to keep track of free seats at rows. Assume some random booking to start with. Use array to store pointers (Head pointer) to each row. On demand

a) The list of available seats is to be displayed
b) The seats are to be booked
c) The booking can be cancelled.

Pre-requisite:

- Knowledge of Doubly Circular Linked List
- Representation of Circular Linked list
- Knowledge of ticket booking

Objective:

- To perform Doubly Circular linked list for cinemax ticket booking.
- To display available seats.
- To book and cancel seats

Input:

Row no and seat no to book seat

Outcome:

- Diplay available seats to book movie ticket.
- Display status of Booked seat/ cancel seat.

Theory :

Circular Doubly linked list : (write CDLL theory in details i.e. definition, concepts, advantages, disadvantages.)

- **Doubly Circular linked list as an ADT : (write pseudo code for each operation)**
- **Algorithms :**

  (Write your own algorithms for your program)

- **Flowchart :**

  (draw flowchart for above algorithms)

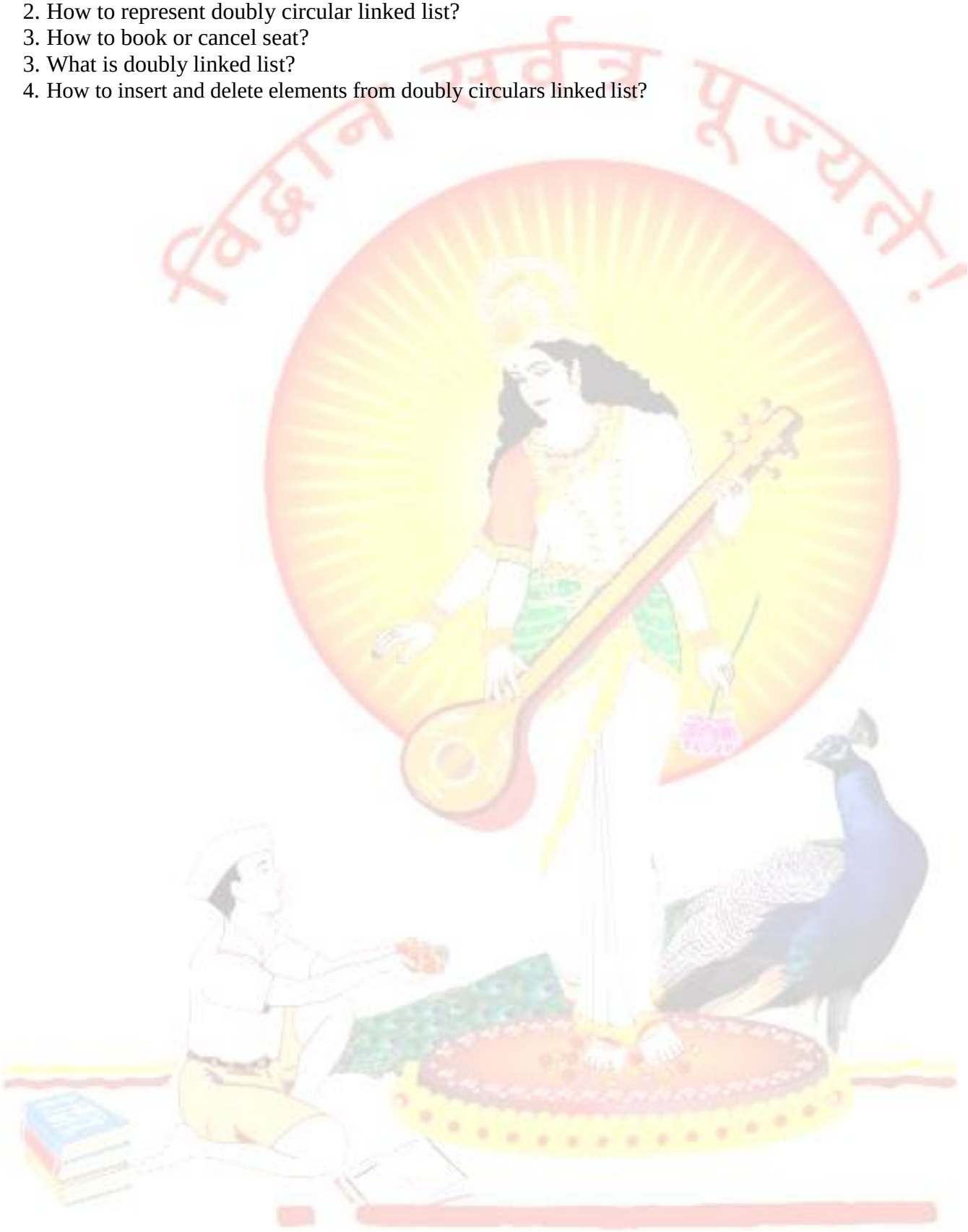
Conclusion:

By this way, we can book or cancel movie ticket using doubly Circular linked lists.

A	P	J	Total	Dated Sign
3	4	3	10	

Questions: **(No need to write answers)**

1. What is Doubly Circular Linked List?
2. How to represent doubly circular linked list?
3. How to book or cancel seat?
3. What is doubly linked list?
4. How to insert and delete elements from doubly circulars linked list?



Practical No:10 (C)

Practical Title: Write C++ program for storing appointment schedule for day. Appointments are booked randomly using **linked list**. Set start and end time and min and max duration for visit slot.

Write functions for

a)Display free slots
b) Book appointment
c) Cancel appointment (check validity, time bounds, availability etc)
d) Sort list based on time
e) Sort list based on time using pointer

Pre-requisite:

- Basics of Singly Linked List
- Different Operations that can be perforemd on Singly linked list

Objective:

- To book or cancel appointment using linked list.

Input:

Start time, end time, min and max time
Status as booked or free

Outcome:

- Display Appointment schedule
- Shoe appointment booking status
- Result of Sort appointment as per start time on linked list
-

Theory : write short theory of linked list.

Explain logic/algorithms to book and cancel appointment.

Explain logic/algorithms for sorting appointment linked list.

Algorithms:

(Write your algorithms for your program.)

Flowchart :

(Draw flowchart for above algorithms.)

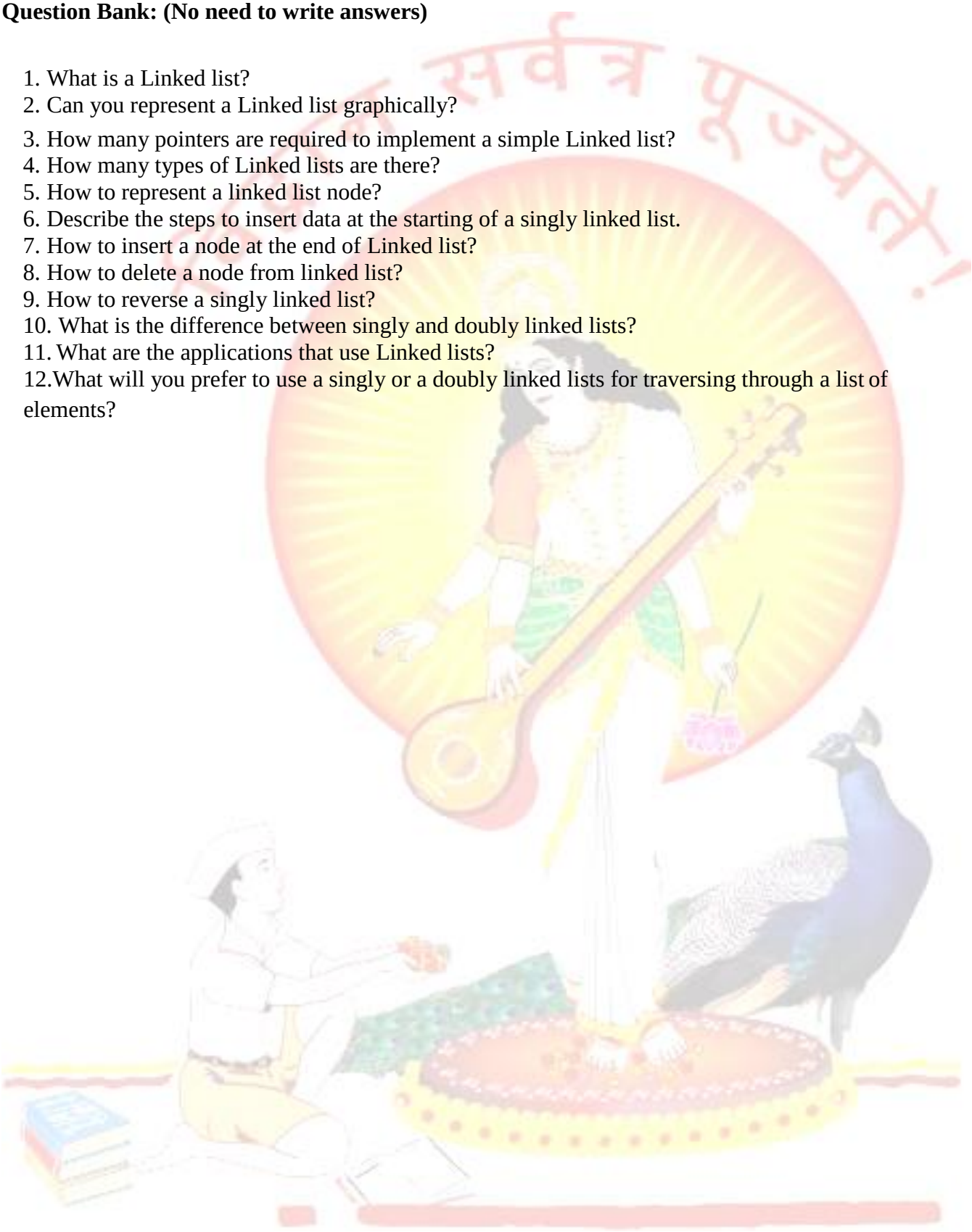
Conclusion:

By this way, we can book or cancel appointment ss using linked list

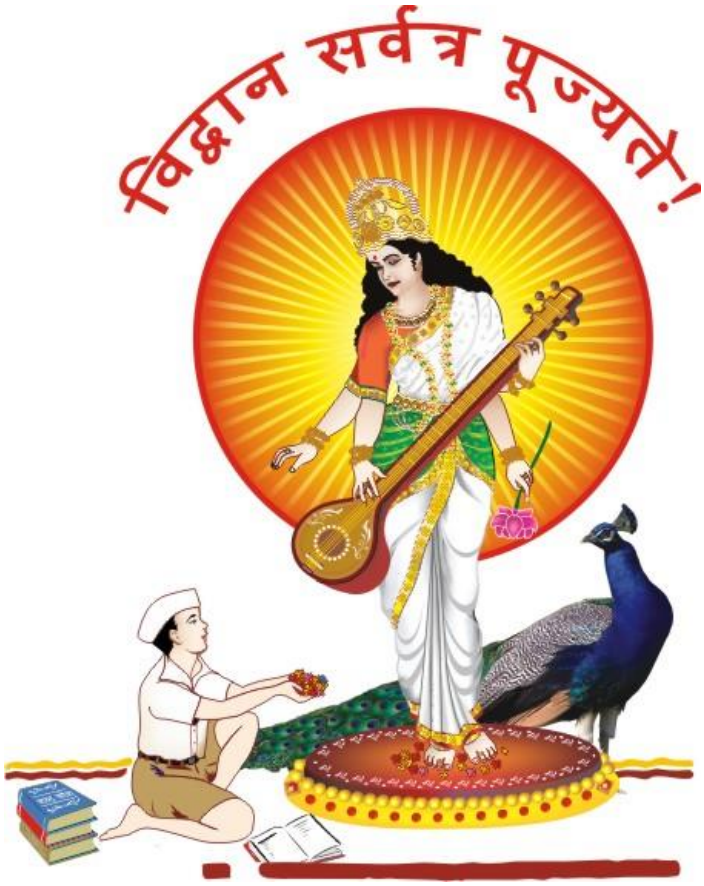
A	P	J	Total	Dated Sign
3	4	3	10	

Question Bank: (No need to write answers)

1. What is a Linked list?
2. Can you represent a Linked list graphically?
3. How many pointers are required to implement a simple Linked list?
4. How many types of Linked lists are there?
5. How to represent a linked list node?
6. Describe the steps to insert data at the starting of a singly linked list.
7. How to insert a node at the end of Linked list?
8. How to delete a node from linked list?
9. How to reverse a singly linked list?
10. What is the difference between singly and doubly linked lists?
11. What are the applications that use Linked lists?
12. What will you prefer to use a singly or a doubly linked lists for traversing through a list of elements?



GROUP - D



GROUP - D

Practical No:11(D)

Practical Title: Write C++ program to check well formedness of parenthesis using stack.

Aim: In any language program mostly syntax error occurs due to unbalancing delimiter such as (), {},[]. Write C++ program using stack to check whether given expression is well parenthesized or not.

Pre-requisite:

- Basics of stack.
- Different operations that can be performed on stack

Objective:

- To check whether the given expression is well parenthesized or not.

Input:

Expresstion using {},(),[].

Outcome:

- Result of checking well formedness of parenthesis.

Theory :

- **Write short theory of stack.**
- **Write concept of well form parenthesis.**
- **Example of well form parenthesis.**

Algorithms :
Write your own algorithms

Flowchart :
Draw flowchart for above algorithms.

Conclusion:

By this way, we can check well formedness of parenthesis using stack.

A	P	J	Total	Dated Sign
3	4	3	10	

Question Bank:

1. What is Stack?
2. Which are the different operations that can be performed on stack?
3. Explain PUSH, POP operations on stack
4. What are the applications of stack?

Practical No:12(D)

Practical Title: Write a C++ program for expression conversion as **infix to postfix** and its evaluation using stack

Aim: Implement C++ program for expression conversion as infix to postfix and its evaluation using stack based on given conditions

i. Operands and operator, both must be single character.

ii. Input Postfix expression must be in a desired format.

iii. Only '+', '-', '*' and '/ ' operators are expected

Pre-requisite:

- Basics of stack.
- Different operations that can be performed on stack

Objective:

- To convert the expression from infix to postfix
- Evaluate the expression

Input:

Infix expression

Outcome:

- Equivalent postfix expression
- Result of evaluation of an expression.

Theory :

- **Write short theory for stack.**
- **Explain infix to postfix expression**
- **Example infix to postfix conversion**

Algorithms :
Write your own algorithms

Flowchart :
Draw flowchart for above algorithms

Conclusion:

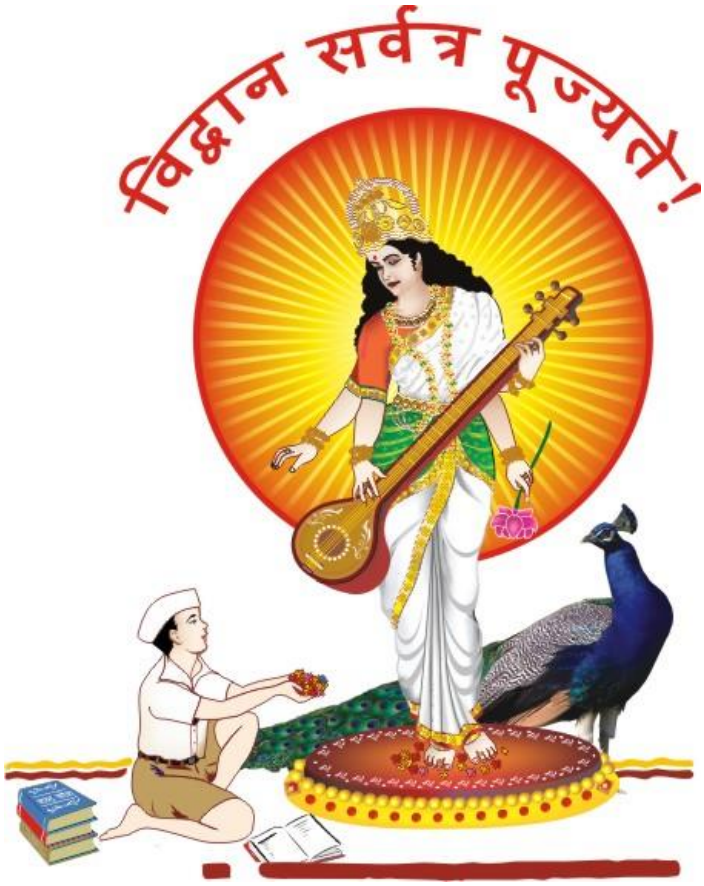
By this way, we can perform expression conversion as infix to postfix and its evaluation using stack

A	P	J	Total	Dated Sign
3	4	3	10	

Question Bank:

1. What is Stack?
2. Which are the different operations that can be performed on stack?
3. Explain PUSH, POP operations on stack
4. What are the applications of stack?
5. What is infix, postfix and prefix expression?
6. Conversion – infix to postfix, infix to prefix , etc.
7. Evaluation of infix, postfix and prefix expression

GROUP - E



GROUP - E

Practical No:13 (E)

Practical Title: Perform different operations on Queue.

Aim: Queues are frequently used in computer programming, and a typical example is the creation of a job queue by an operating system. If the operating system does not use priorities, then the jobs are processed in the order they enter the system. Write C++ program for simulating job queue.Write functions to add job and delete job from queue.

Pre-requisite:

- Basics of Queue
- Different operations that can be performed on queue

Objective:

- To perform addition and deletion operations on queue.

Input:

Size of queue

Elements in ququе

Outcome:

- Result of addition of job operation on queue.
- Result of deletion of job operation on queue.

Theory :

- **Write theory of queue (definition, concepts, types, advantages, disadvantages)**
- **Explain queue as an ADT. (write pseudo code)**

Algorithms :

Step 1: Include all the header files which are used in the program and define a constant 'SIZE' with specific value.
Step 2: Declare all the user defined functions which are used in queue implementation.
Step 3: Create a one dimensional array with above defined SIZE (int queue[SIZE])
Step 4: Define two integer variables 'front' and 'rear' and initialize both with '-1'. (int front = -1, rear = -1)
Step 5: Then implement main method by displaying menu of operations list and make suitable function calls to perform operation selected by the user on queue.

enQueue(value) - Inserting value into the queue:

In a queue data structure, enQueue() is a function used to insert a new element into the queue. In a queue, the new element is always inserted at rear position. The enQueue() function takes one integer value as parameter and inserts that value into the queue. We can use the following steps to insert an element into the queue...

Step 1: Check whether queue is FULL. (rear == SIZE-1)
Step 2: If it is FULL, then display "Queue is FULL!!! Insertion is not possible!!!" and terminate the function.
Step 3: If it is NOT FULL, then increment rear value by one (rear++) and set queue[rear] = value.

deQueue() - Deleting a value from the Queue:

In a queue data structure, deQueue() is a function used to delete an element from the queue. In a queue, the element is always deleted from front position. The deQueue() function does not take any value as parameter. We can use the following steps to delete an element from the queue...

Step 1: Check whether queue is EMPTY. (front == rear)
Step 2: If it is EMPTY, then display "Queue is EMPTY!!! Deletion is not possible!!!" and terminate the function.
Step 3: If it is NOT EMPTY, then increment the front value by one (front ++). Then display queue[front] as deleted element. Then check whether both front and rear are equal (front == rear), if it TRUE, then set both front and rear to '-1' (front = rear = -1).

display() - Displays the elements of a Queue:
We can use the following steps to display the elements of a queue...

Step 1: Check whether queue is EMPTY. (front == rear)
Step 2: If it is EMPTY, then display "Queue is EMPTY!!!" and terminate the function.
Step 3: If it is NOT EMPTY, then define an integer variable 'i' and set 'i = front+1'.
Step 3: Display 'queue[i]' value and increment 'i' value by one (i++). Repeat the same until 'i' value is equal to rear (i <= rear)

Flowchart :
Draw flowchart for above algorithms

Conclusion:

By this way, we can perform different operations on queue

A	P	J	Total	Dated Sign
3	4	3	10	

Question Bank:
1. What is Queue?
2. What are the different operations that can be performed on queue?
3. Explain all the operations on queue
4. Which are different types of queues , Explain.

Practical No:14 (E)

Practical Title: Perform operations on Double ended queue.

Aim: A double-ended queue(deque) is a linear list in which additions and deletions may be made at either end. Obtain a data representation mapping a deque into a one-dimensional array. Write C++ program to simulate deque with functions to add and delete elements from either end of the deque.

Pre-requisite:

- Knowledge of Queue
- Types of queue
- Knowledge of double ended queue and different operations that can be performed on it

Objective:

- To simulate deque with functions to add and delete elements from either end of the deque.

Input:

Size of array

Elements in the queue

Outcome:

- Result of deque with functions to add and delete elements from either end of the deque.

Theory :

Double-Ended Queue

A double-ended queue is an abstract data type similar to an simple queue, it allows you to insert and delete from both sides means items can be added or deleted from the front or rear end.

Algorithm for Insertion at rear end

Step -1: [Check for overflow]
if(rear==MAX)
Print("Queue is Overflow");
return;
Step-2: [Insert element]
else
rear=rear+1;

```
    q[rear]=no;
    [Set rear and front pointer]

  if rear=0
    rear=1;
  if front=0
    front=1;
  Step-3: return
```

Implementation of Insertion at rear end

```
  void add_item_rear()
  {
    int num;
    printf("\n Enter Item to insert : ");
    scanf("%d",&num);
    if(rear==MAX)
    {
    printf("\n Queue is Overflow");
    return;
   }
    else
    {
   rear++;
   q[rear]=num;
   if(rear==0)
   rear=1;
   if(front==0)
   front=1;
  }
   }
```

Algorithm for Insertion at font end

```
  Step-1 : [Check for the front position]
   if(front<=1)
   Print ("Cannot add item at front end");
   return;
  Step-2 : [Insert at front]
   else
  front=front-1;
  q[front]=no;
  Step-3 : Return
```

Implementation of Insertion at font end

```
  void add_item_front()
  {
  int num;
  printf("\n Enter item to insert:");
   scanf("%d",&num);
```

```
 if(front<=1)

{
 printf("\n Cannot add item at front end");
return;
 }
 else
 {
 front--;
 q[front]=num;
}
}
```

Algorithm for Deletion from front end

```
Step-1 [ Check for front pointer]
 if front=0
 print(" Queue is Underflow");
  return;
   Step-2 [Perform deletion]
  else
    no=q[front];
 print("Deleted element is",no);
 [Set front and rear pointer]
 if front=rear
 front=0;
 rear=0;
 else
 front=front+1;
 Step-3 : Return
```

Implementation of Deletion from front end

```
 void delete_item_front()
 {
 int num;
 if(front==0)
 {
 printf("\n Queue is Underflow\n");
 return;
 }
 else
 {
 num=q[front];
 printf("\n Deleted item is %d\n",num);
 if(front==rear)
 {
 front=0;
 rear=0;
 }
 else
 {
 front++;
```

```
    }

    }
}
```

Algorithm for Deletion from rear end

Step-1 : [Check for the rear pointer]
if rear=0
print("Cannot delete value at rear end");
return;
Step-2: [perform deletion]
else
no=q[rear];
[Check for the front and rear pointer]
if front= rear
front=0;
rear=0;
else
rear=rear-1;
print("Deleted element is",no);
Step-3 : Return

Implementation of Deletion from rear end

```
   void delete_item_rear()
 {
   int num;
   if(rear==0)
   {
   printf("\n Cannot delete item at rear end\n");
   return;
   }
   else
   {
   num=q[rear];
   if(front==rear)
   {
   front=0;
   rear=0;
   }
   else
   {
   rear--;
   printf("\n Deleted item is %d\n",num);
   }
   }
  }
```

Algorithms :
Write your own algorithms

Flowchart :
Draw flowchart for above algorithms

Conclusion:

By this way, we can perform operations on double ended queue

A	P	J	Total	Dated Sign
3	4	3	10	

Question Bank:

1. What is queue?
2. Types of queue
3. What is double ended queue?
4. How to insert the new node in Doubly Link List?
5. How to delete the node from front of Doubly Link List ?
6. How to delete the node from end of Doubly Link List ?
7. How to delete the node in between of Doubly Link List

Practical No:15 (E)

Practical Title: Perform operations on Circular queue.

Aim: Pizza parlor accepting maximum M orders. Orders are served in first come first served basis. Order once placed cannot be cancelled. Write C++ program to simulate the system using **circular queue** using array

Pre-requisite:

- Knowledge of Circular Queue
- Types of Circular queue
- Knowledge of Singly and double Circular queue and different operations.

Objective:

- To simulate circular queue with functions to add and delete elements from Circular queue.

Input:

Accept order

Elements in the queue

Outcome:

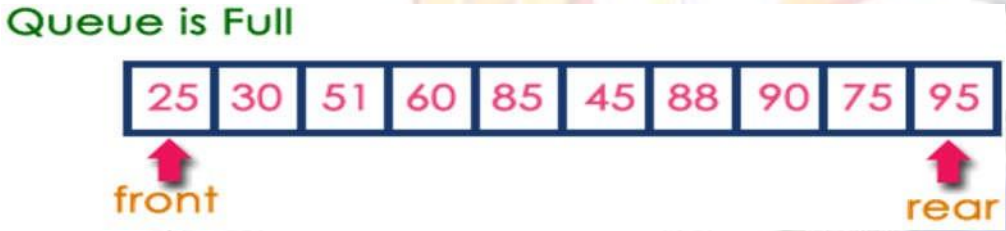
- Result of Circular queue with functions to accept or cancel Pizza order.

Theory :

- **Why Circular Queue?**

In a normal Queue Data Structure, we can insert elements until queue becomes full. But once if queue becomes full, we can't insert the next element until all the elements are deleted from the queue. For example consider the queue below...

After inserting all the elements into the queue…



Now consider the following situation after deleting three elements from the queue...

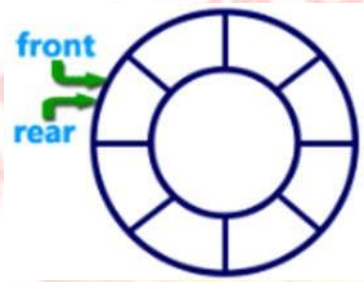
This situation also says that Queue is Full and we can't insert the new element because, 'rear' is still at last position. In above situation, even though we have empty positions in the queue we can't make use of them to insert new element. This is the major problem in normal queue data structure. To overcome this problem we use circular queue data structure.

- **What is a Circular Queue?**

A Circular Queue can be defined as follows...

Circular Queue is a linear data structure in which the operations are performed based on FIFO (First In First Out) principle and the last position is connected back to the first position to make a circle.

Graphical representation of a circular queue is as follows...



Implementation of Circular Queue:

To implement a circular queue data structure using array, we first perform the following steps before we implement actual operations.

Step 1: Include all the header files which are used in the program and define a constant 'SIZE' with specific value.

Step 2: Declare all user defined functions used in circular queue implementation.

Step 3: Create a one dimensional array with above defined SIZE (int cQueue[SIZE])

Step 4: Define two integer variables 'front' and 'rear' and initialize both with '-1'. (int front = -1, rear = -1) Step 5: Implement main method by displaying menu of operations list and make suitable function calls to perform operation selected by the user on circular queue.

enQueue(value) - Inserting value into the Circular Queue:

In a circular queue, enQueue() is a function which is used to insert an element into the circular queue. In a circular queue, the new element is always inserted at rear position. The enQueue() function takes one integer value as parameter and inserts that value into the circular queue. We can use the following steps to insert an element into the circular queue...

Step 1: Check whether queue is FULL. ((rear == SIZE-1 && front == 0) || (front == rear+1))

Step 2: If it is FULL, then display "Queue is FULL!!! Insertion is not possible!!!" and terminate the function.

Step 3: If it is NOT FULL, then check rear == SIZE - 1 && front != 0 if it is TRUE, then set rear = -1.

Step 4: Increment rear value by one (rear++), set queue[rear] = value and check 'front == -1' if it is TRUE, then set front = 0.

deQueue() - Deleting a value from the Circular Queue:

In a circular queue, deQueue() is a function used to delete an element from the circular queue. In a circular queue, the element is always deleted from front position. The deQueue() function doesn't take any value as parameter. We can use the following steps to delete an element from the circular queue...

Step 1: Check whether queue is EMPTY. (front == -1 && rear == -1)
Step 2: If it is EMPTY, then display "Queue is EMPTY!!! Deletion is not possible!!!" and terminate the function.
Step 3: If it is NOT EMPTY, then display queue[front] as deleted element and increment the front value by one (front ++). Then check whether front == SIZE, if it is TRUE, then set front = 0. Then check whether both front - 1 and rear are equal (front -1 == rear), if it TRUE, then set both front and rear to '-1' (front = rear = -1).

display() - Displays the elements of a Circular Queue:

We can use the following steps to display the elements of a circular queue...

Step 1: Check whether queue is EMPTY. (front == -1)
Step 2: If it is EMPTY, then display "Queue is EMPTY!!!" and terminate the function.
Step 3: If it is NOT EMPTY, then define an integer variable 'i' and set 'i = front'.
Step 4: Check whether 'front <= rear', if it is TRUE, then display 'queue[i]' value and increment 'i' value by one (i++). Repeat the same until 'i <= rear' becomes FALSE.
Step 5: If 'front <= rear' is FALSE, then display 'queue[i]' value and increment 'i' value by one (i++). Repeat the same until'i <= SIZE - 1' becomes FALSE.
Step 6: Set i to 0.
Step 7: Again display 'cQueue[i]' value and increment i value by one (i++). Repeat the same until 'i <= rear' becomes FALSE.

Algorithms :
Write your own algorithms

Flowchart :
Draw flowchart for above algorithms

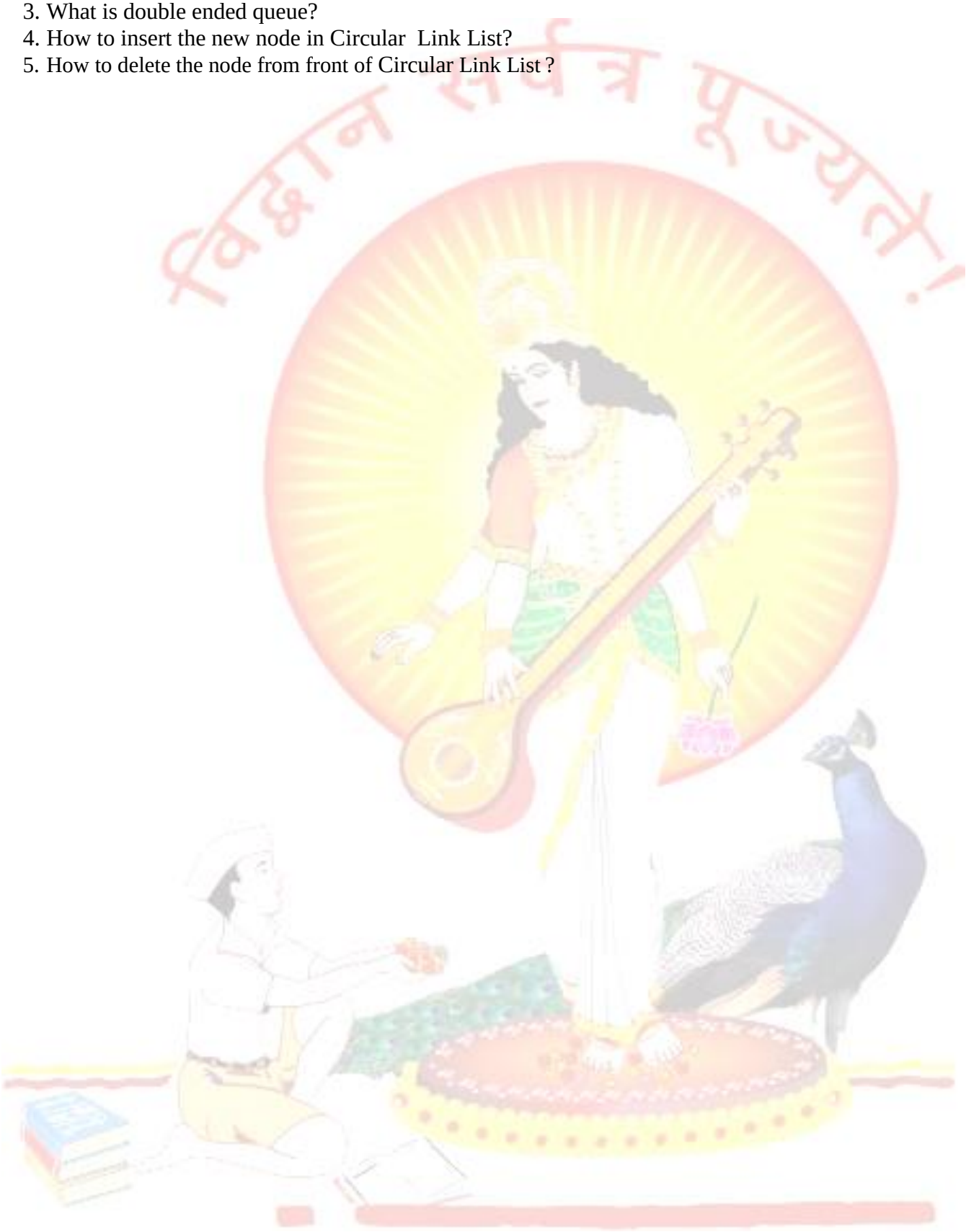
Conclusion:

Hence we have learned how to implement Circular Queue and perform various operations on it.

A	P	J	Total	Dated Sign
3	4	3	10	

Question Bank:

1. What is Circular queue?
2. Types of Circular queue
3. What is double ended queue?
4. How to insert the new node in Circular Link List?
5. How to delete the node from front of Circular Link List ?



THANKS..!

PROF. ANAND NANDLAL GHARU

ASSISTANT PROFESSOR

PVGCOE, NASHIK

Blog : anandgharu.wordpress.com