Pune Vidyarthi Griha's

COLLEGE OF ENGINEERING, NASIK

SE COMPUTER ENGINEERING DEPT.

LAB MANUAL

DATA STRUCTURE LABORATORY

Prepared by

PROF. ANAND GHARU

2019 - 20

PUNE VIDYARTHI GRIHA'S

COLLEGE OF ENGINEERING,NASHIK.

INDEX Batch : -

Sr. No	Title	Page No	Date of Conduction	Date of Submission	Signre of Staff
	GROUP - A				
1	Write C/C++ program to store marks scored for first test of subject 'Data Structures and Algorithms' for N students. Compute I. The average score of class ii. Highest score and lowest score of class iii. Marks scored by most of the students iv. list of students w ho were absent for the test				
2	Set A=(1,3, a, s, t, i} represent alphanumeric Characters permitted to set the password of length 4. Write C/C++ program to generate all possible passwords				
3	Write C/C++ program for storing matrix. Write functions for a) Check whether given matrix is upper triangular/not b) Compute summation of diagonal elements c) Compute transpose of matrix d) Add, subtract and multiply two matrices Write C++ program for string operations- copy, concatenate, check substring, equal, reverse and length				
4	matricesWrite C++ program for string operations- copy, concatenate, check substring, equal, reverse and length				
	GROUP - B				
5	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) Display list in reverse order using recursion e) Two linked lists exists for two divisions. Concatenate.				

6	Write C++ program for storing binary number using doubly linked lists. Write functions a) to compute 1's and 2's complement b) add two binary numbers				
7	Write C++ program to store set of negative and positive numbers using linked list. Write functions to a) Insert numbers b) Delete nodes with negative numbers c) Create two more linked lists using this list, one containing all positive numbers and other containing negative numbers d) For two lists that are sorted; Merge these two lists into third resultant list that is sorted				
	GROUP - C				
8	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.				
9	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.				
	GROUP - D				
10	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.				
11	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.				

PUNE VIDYARTHI GRIHA'S

COLLEGE OF ENGINEERING,NASHIK.

INDEX Batch : -

	GROUP - E				
12	Write C++ program to store first year percentage of students in array. Write function for sorting array of floating point numbers in ascending order using a) Selection Sort b) Bubble sort and display top five scor				
13	Write C++ program to store first year percentage of students in array. Sort array of floating point numbers in ascending order using quick sort and display top five scores .				
14	Write C++ 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 `				

Certified that Mr/Miss _____ of

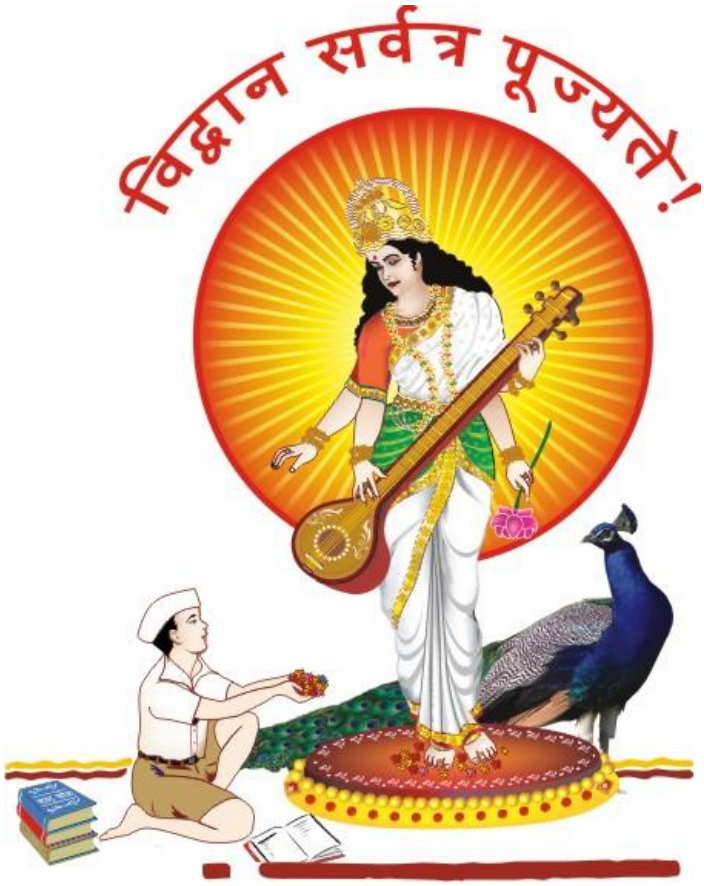
class _____ Sem ____ Roll no. ____ has completed the term work satisfactorily in the

subject _____of the Department _____of

PVG's College of Engineering Nashik. During academic year _____ .

Staff Member	**Head of Dept.**	**Principal**
Prof. Gharu A. N.	_____	_____

GROUP - A



GROUP - A

DATA STRUCTURES LAB
(210247)

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.

Operating System recommended :- 64-bit Open source Linux or its derivative
Programming tools recommended: - Open Source C++ Programming tool like G++/GCC

INTRODUCTION

This manual is organized in such a way that the students can directly use it in the laboratory. Each laboratory exercise comprises of

1. Statement of the problem
2. Algorithm
3. C code
4. Results
5. Conclusion

You must follow this sequence to conduct any data structures experiment in the computer laboratory and write your report accordingly. Using the algorithms given in this section one can implement in any language at a later time. For example, to implement the stack program in C++ or Java (of course now you must use C language) simply follow the same algorithmic steps and use additional language features, if any. This is the reason why the students must write the algorithm first and then the C code.

INTRODUCTION TO C

C was brought to us by Dennis Ritchie and Brian at Bell Labs. AT&T people had themselves this Operating System called UNIX that needed a programming language. They made it, and since the previous version was called B, they decided to call it C for obvious reasons. There was never an A programming language. The B in B stood for Bell. C is a computer programming language. That means that you can use C to create lists of instructions for a computer to follow. C is one of thousands of programming languages currently in use. C has been around for several decades and has won widespread acceptance because it gives programmers maximum control and efficiency. C is what is called a compiled language. This means that once you write your C program, you must run it through a C compiler to turn your program into an executable that the computer can run (execute). The C program is the human-readable form, while the executablethat comes out of the compiler is the machine-readable and executable form. What this means is that to write and run a C program, you must have access to a C compiler. If you are using a UNIX machine (for example, if you are writing CGI scripts in C on your host's UNIX computer, or if you are a student working on a lab's UNIX machine), the C compiler is available for free. It is called either "cc" or "gcc" and is available on the command line.

CHARACTERISTICS OF 'C' LANGUAGE

Modularity:
Ability to breakdown a large module into manageable sub modules called as modularity that is an important feature of structured programming languages.

Advantages:

1. Projects can be completed in time.
2. Debugging will be easier and faster.

Portability:
The ability to port i.e. to install the software in different platform is called portability. Highest degree of portability: 'C' language offers highest degree of portability i.e., percentage of changes to be made to the sources code is at minimum when the software is to be loaded in another platform.

Percentage of changes to the source code is minimum.

Extendability:
Ability to extend the existing software by adding new features is called as extendability.

SPEED:
'C' is also called as middle level language because programs written in 'c' language run at the speeds matching to that of the same programs written in assembly language so 'c' language has both the merits of high level and middle level language and because if this feature it is mainly used in developing system software.

AREAS OF APPLICATIONS

The C programming language is used in many different areas of application, but the most prolific area is UNIX operating system applications. The C language is also used in computer games:
• UNIX operating system
• Computer games

DATA STRUCTURES

A data structure is an arrangement of data in a computer's memory or even disk storage. An example of several common data structures are arrays, linked lists, queues, stacks, binary trees, and hash tables. Algorithms, on the other hand, are used to manipulate the data contained in these data structures as in searching and sorting. C provides us with just one built in data structure, the array. Although efficient and easily used and understood, arrays often don't provide us with the level of functionality we need from a data structure. If the quantity of data we need to store is not well known at compile time, then using arrays could waste memory if too large, or waste time in resizing at runtime if too small. Several, more abstract, data structures can be constructed to better serve our needs. Many algorithms apply directly to a specific data structures. When working with certain data structures you need to know how to insert new data, search for a specified item, and deleting a specific item. Commonly used algorithms include are useful for:
• Searching for a particular data item (or record).
• Sorting the data. There are many ways to sort data. Simple sorting, Advanced sorting
• Iterating through all the items in a data structure. (Visiting each item in turn so as to display it or perform some other action on these items)

Software Requirements

- **Operating System :** Windows XP/ ubuntu 14.04 LTS
- **Software :**Turbo c/c++, gcc, g++, eclipse, Editor

Hardware Requirements

- **Processor :** Intel® Core™ i3-3220 CPU @ 3.30GHz × 4
- **RAM :** 256 MB
- **HDD :** 40 GB

Practical No.	Laboratory Assignments
	GROUP - A
1	Write C/C++ program to store marks scored for first test of subject 'Data Structures and Algorithms' for N students. Compute I. The average score of class ii. Highest score and lowest score of class iii. Marks scored by most of the students iv. list of students who were absent for the test
2	Set A=(1,3, a, s, t, i} represent alphanumeric characters permitted to set the password of length 4. Write C/C++ program to generate all possible passwords
3	Write C/C++ program for storing matrix. Write functions for a) Check whether given matrix is upper triangular or not b) Compute summation of diagonal elements c) Compute transpose of matrix d) Add, subtract and multiply two matricesWrite C++ program for string operations- copy, concatenate, check substring, equal, reverse and length
4	matricesWrite C++ program for string operations- copy, concatenate, check substring, equal, reverse and length
	GROUP - B
5	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) Display list in reverse order using recursion e) Two linked lists exists for two divisions. Concatenate two lists.
6	Write C++ program for storing binary number using doubly linked lists. Write functions a) to compute 1‘s and 2‘s complement b) add two binary numbers
7	Write C++ program to store set of negative and positive numbers using linked list. Write functions to a) Insert numbers b) Delete nodes with negative numbers c) Create two more linked lists using this list, one containing all positive numbers and other containing negative numbers d) For two lists that are sorted; Merge these two lists into third resultant list that is sorted
	GROUP - C
8	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.

9	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.
GROUP - D	
10	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.
11	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.
GROUP - E	
12	Write C++ program to store first year percentage of students in array. Write function for sorting array of floating point numbers in ascending order using a) Selection Sort b) Bubble sort and display top five scores.
13	Write C++ program to store first year percentage of students in array. Sort array of floating point numbers in ascending order using quick sort and display top five scores .
14	Write C++ 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

Practical No: 01(A)

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

Aim: Write C/C++ program to store marks scored for first test of subject 'Data Structures and Algorithms' for N students. Compute

I. The average score of class

ii. Highest score and lowest score of class

iii. Marks scored by most of the students

iv. list of students who were absent for the test

Prerequisite:

- C++ Programming

Objectives:

- To understand the use functions for N students record.

Input: N number of students.

Outcome: Resulting average, highest and lowest marks operation.

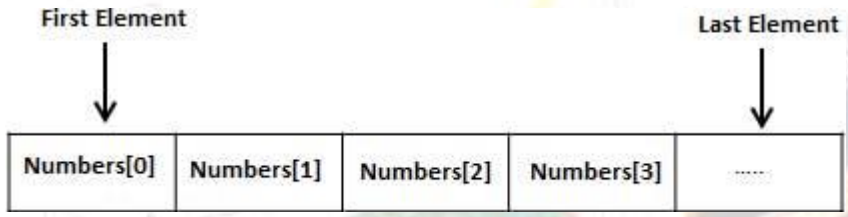
Theory:

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.

1. The average score of class

2. Highest score and lowest score of class

3. Marks scored by most of the students

4. list of students who were absent for the test

Algorithm:

1. Start
2. Accept number of students from user.
3. Accept roll number and marks of student.
4. Write function to calculate average marks
5. Write function to calculate highest marks
6. Write function to calculate lowest marks
7. Display roll number of absent student.
8. Stop.

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	

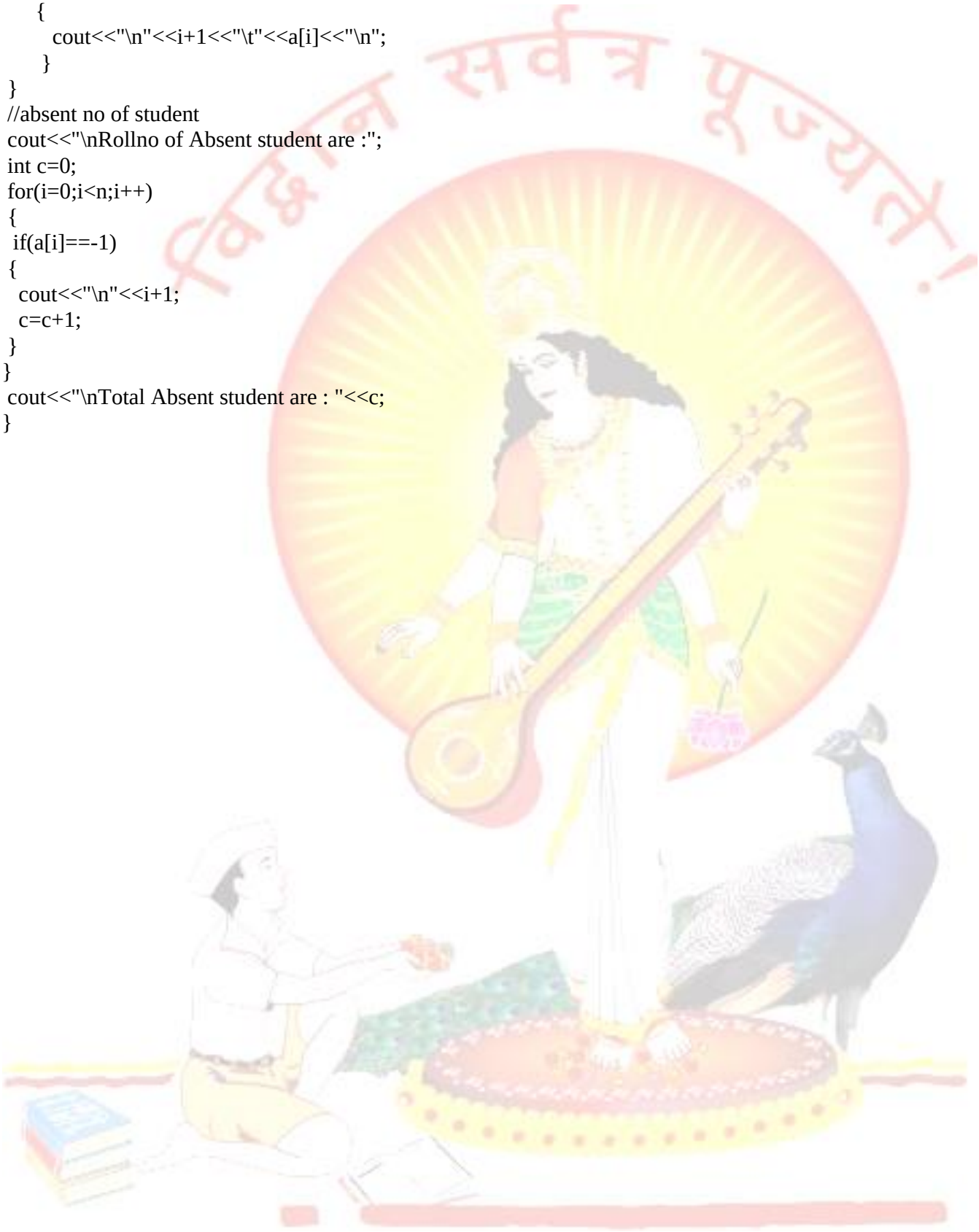
Questions:

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

Program :

```
#include<iostream>
using namespace std;
int main()

{
        int n,max=0,min,i,a[20],sum=0,j;
     float avg;
        cout<<"Enter the Number of Student : ";
     cin>>n;
    cout<<"Enter the marks of Student :";
    cout<<"\nRollno"<<"\t"<<"Marks";
    for(int i=0;i<n;i++)
    {
      cout<<"\n"<< i+1<<" :\t";
      cin>>a[i];
     sum=sum+a[i];
    }
    avg=sum/n;
  cout<<"Average of total student is :";
  cout<<avg<<"\n";
 //MAx value
 for(i=0;i<n;i++)
 {
  if(a[i]>max)
   {
     max=a[i];
     j=i+1;
   }
}

 cout<<"Higest mark is :"<< j<<"\t"<<max<<"\n";

 //min value
 for(i=0;i<n;i++)
 {
 if(a[i]>=0)
 {
  min=a[i];
   break;
 }
}
 for(i=0;i<n;i++)
{
 if(a[i]>0)
 {
   if(a[i]<min)
   {
     min=a[i];
     j=i+1;
   }
 }
 }
  cout<<"lowest mark is :"<< j<<"\t"<<min<<"\n";
```

```
//above 75 mark
 cout<<"Who score more that 75 marks :";
 for(i=0;i<n;i++)
{
    if(a[i]>75)
    {
      cout<<"\n"<<i+1<<"\t"<<a[i]<<"\n";
     }
}
//absent no of student
cout<<"\nRollno of Absent student are :";
int c=0;
for(i=0;i<n;i++)
{
 if(a[i]==-1)
{
  cout<<"\n"<<i+1;
  c=c+1;
}
}
 cout<<"\nTotal Absent student are : "<<c;
}
```



Practical No : 2(A)

Practical Title: Write C/C++ program to generate all possible passwords**.**

Aim : Set A=(1,3, a, s, t, i} represent alphanumeric characters permitted to set the password of length 4. Write C/C++ program to generate all possible passwords

Pre-requisite:

- Knowledge of representing looping in C/C++
- Knowledge of permutation for possible password

Objectives:

- Generate 4 digit password
- Generate password for alphanumeric character .

Input:
Set of alphanumberic character

Outcome :
Generate 4 digit password

Theory :

Definition.

Permutations are the different ways in which a collection of items can be arranged. For **example**: The different ways in which the alphabets A, B and C can be grouped together, taken all at a time, are ABC, ACB, BCA, CBA, CAB, BAC.

Question:

I need all of the 4 digit combinations using the numbers 0-9?

➔ you can repeat the digits the NUMBER of 4 digit combinations is relatively easy to calculate. Suppose that you are going to write one of these 4 digit combinations. You have 10 choices for the first digit. Once you have chosen the first digit you have 10 choices for the second digit. Thus you have 10x10 = 100 choices for the first two digits. For each choice of the first two digits you have 10 choices for the third digit. Thus you have 10x10x10 = 1000 choices for the first three digits. Finally you have 10 choices for the fourth digit and thus there are **10x10x10x10 = 10 000** possible **4 digit combinations from 0-9**.

I am not going to list all 10 000 combinations but I will show you a way to do so in a "natural" order. The list will be in 1000 rows with 10 combinations in each row.

Example :

0000 0001 0002 0003 0004 ... 0008 0009

0010 0011 0012 0013 0014 ... 0018 0019

0020 0021 0022 0023 0024 ... 0028 0029

. . . .
. . . .
. . . .
. . . .

9980 9981 9982 9983 9984 ... 9988 9989

9990 9991 9992 9993 9994 ... 9998 9999

Algorithms :

1. Start
2. Take input as set of alphanumeric character.
3. Write function to generate password
   - Define nos of for loop as per requirement.(nested loop)
   - Repeat for loop till displaying all password
4. Display password
5. End

Flowchart :

Here, draw flowchart as per your algorithms

Conclusion :

By this way, we can generate 4 digit password sucessfully.

A	**P**	**J**	**Total**	**Dated Sign**
3	4	3	10	

Program :

```
#include<iostream>
using namespace std;
class password

{
char seta[6]={'1','2','a','s','t','i'};
int count=0;

public:

void combination();
};

void password::combination()

{

for(int i=0;i<6;i++)
{

for(int j=0;j<6;j++)
{

for(int k=0;k<6;k++)
{

for(int l=0;l<6;l++)
{

cout<<seta[i]<<seta[j]<<seta[k]<<seta[l];
cout<<"\t";
  count++;
}
}
}
}
cout<<"\nTOTAL COMBINATION ARE:"<<count;
}

int main()
{
password p;
p.combination();
}
```

////**OUTPUT :**

```
1111   1112   111a   111s   111t   111i   1121   1122   112a   112s   112t   112i   11a1   11a2
       11aa   11as   11at   11ai   11s1   11s2   11sa   11ss   11st   11si   11t1   11t2   11ta
       11ts   11tt   11ti   11i1   11i2   11ia   11is   11it   11ii   1211   1212   121a   121s
       121t   121i   1221   1222   122a   122s   122t   122i   12a1   12a2   12aa   12as   12at
       12ai   12s1   12s2   12sa   12ss   12st   12si   12t1   12t2   12ta   12ts   12tt   12ti
       12i1   12i2   12ia   12is   12it   12ii   1a11   1a12   1a1a   1a1s   1a1t   1a1i   1a21
       1a22   1a2a   1a2s   1a2t   1a2i   1aa1   1aa2   1aaa   1aas   1aat   1aai   1as1   1as2
```

Practical No: 3(A)

Practical Title: Perform different operations on Matrix.

Aim : Write C/C++ program for storing matrix. Write functions for
b) Check whether given matrix is upper triangular or not
c) Compute summation of diagonal elements
d) Compute transpose of matrix
e) Add, subtract and multiply two matrices

Pre-requisite:

- Knowledge of representing matrix in C/C++
- Knowledge of different operations that can be performed on matrix

Objectives:

- Check whether the the matrix is upper triangular or not
- Compute the transpose and summation of diagonal elements 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
- Whether the matrix is upper triangular
- 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**
  - **Upper traingular matrix**
  - **Transpose of matrix**
  - **Summation of diagonal 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. Check the elements of first matrix, if all the entries below the main diagonal are zero then output that it is upper triangular matrix.
7. Write function to compute summation of diagonal elements of both the matrices.
8. 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.
9. 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:

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

Program :

```
#include<iostream>
using namespace std;
int main()
{       char chr;
        int a[10][10],b[10][10],c[10][10],row,col,i,j,ch,m=0,p=0,q=0;
        do{
        cout<<"\t"<<"*************************"<<endl<<endl;
        cout<<"\t"<<"ENTER YOUR CHOICE NUMBER"<<endl<<endl<<endl;
        cout<<"\t"<<"_______MENU_______"<<endl<<endl;
        cout<<"\t"<<"1).MATRIX ADDITION"<<endl;
        cout<<"\t"<<"2).MATRIX SUBSTRACTION"<<endl;
        cout<<"\t"<<"3).MATRIX MULTIPLICATION"<<endl;
        cout<<"\t"<<"4).MATRIX TRANSPOSE"<<endl;
        cout<<"\t"<<"5).SUMMATION OF DIAGONAL ELEMENT"<<endl;
        cout<<"\t"<<"6).CHECK UPPER TRIANGULAR MATRIX"<<endl;
        cout<<"\t"<<"7).EXIT"<<endl;
        cout<<"\t"<<"*************************"<<endl<<endl;

        cin>>ch;   // acceppt choice from user

        switch(ch)
        {
                case 1: //addition of matrix
                                // matrix a
                                cout<<"ENTER MATRIX 'A' "<<endl;
                                cout<<"ENTER NO OF ROWS OF MATRIX A"<<endl;
                                cin>>row;
                                cout<<"ENTER NO OF COLOUMNS OF MATRIX A"<<endl;
                                cin>>col;
                                cout<<"ENTER MATRIX ELEMENTS :"<<endl;
                                for(i=0;i<row;i++)
                                {
                                        for(j=0;j<col;j++)
                                        {
                                                cout<<"\t"<<"a["<<i<<"]["<<j<<"]\t";
                                                cin>>a[i][j];
                                        }

                                }
                                cout<<endl<<endl<<"YOUR MATRIX A IS :"<<endl;
                                for(i=0;i<row;i++)
                                {    cout<<"||"<<"\t";
                                        for(j=0;j<col;j++)
                                        {
                                                cout<<a[i][j]<<"\t";
                                        }
                                        cout<<"||"<<endl;
                                }
                                //matrix b
                                cout<<"ENTER MATRIX 'B' "<<endl;
                                cout<<"ENTER NO OF ROWS OF MATRIX B"<<endl;
                                cin>>row;
                                cout<<"ENTER NO OF COLOUMNS OF MATRIX B"<<endl;
                                cin>>col;
```

```
cout<<"ENTER MATRIX ELEMENTS :"<<endl;
for(i=0;i<row;i++)
{
        for(j=0;j<col;j++)
        {
                cout<<"\t"<<"b["<<i<<"]["<<j<<"]\t";
                cin>>b[i][j];
        }

}
cout<<endl<<endl<<"YOUR MATRIX B IS :"<<endl;
for(i=0;i<row;i++)
{   cout<<"||"<<"\t";
        for(j=0;j<col;j++)
        {
                cout<<b[i][j]<<"\t";
        }
        cout<<"||"<<endl;
}
//ADDITION OF MATRIX A & B   .
for(i=0;i<row;i++)
{
        for(j=0;j<col;j++)
        {
                c[i][j]=a[i][j]+b[i][j];
        }

}
cout<<endl<<endl<<"YOUR MATRIX ADDITION IS :"<<endl;
for(i=0;i<row;i++)
{   cout<<"||"<<"\t";
        for(j=0;j<col;j++)
        {
                cout<<c[i][j]<<"\t";
        }
        cout<<"||"<<endl;
}
break;

case 2: //substraction of matrix
        // matrix a
        cout<<"ENTER MATRIX 'A' "<<endl;
        cout<<"ENTER NO OF ROWS OF MATRIX A"<<endl;
        cin>>row;
        cout<<"ENTER NO OF COLOUMNS OF MATRIX A"<<endl;
        cin>>col;
        cout<<"ENTER MATRIX ELEMENTS :"<<endl;
        for(i=0;i<row;i++)
        {
                for(j=0;j<col;j++)
                {
                        cout<<"\t"<<"a["<<i<<"]["<<j<<"]\t";
                        cin>>a[i][j];
                }

        }
        cout<<endl<<endl<<"YOUR MATRIX A IS :"<<endl;
```

```
for(i=0;i<row;i++)
{    cout<<"||"<<"\t";
        for(j=0;j<col;j++)
        {
                cout<<a[i][j]<<"\t";
        }
        cout<<"||"<<endl;
}
//matrix b
cout<<"ENTER MATRIX 'B' "<<endl;
cout<<"ENTER NO OF ROWS OF MATRIX B"<<endl;
cin>>row;
cout<<"ENTER NO OF COLOUMNS OF MATRIX B"<<endl;
cin>>col;
cout<<"ENTER MATRIX ELEMENTS :"<<endl;
for(i=0;i<row;i++)
{
        for(j=0;j<col;j++)
        {
                cout<<"\t"<<"b["<<i<<"]["<<j<<"]\t";
                cin>>b[i][j];
        }

}
cout<<endl<<endl<<"YOUR MATRIX B IS :"<<endl;
for(i=0;i<row;i++)
{    cout<<"||"<<"\t";
        for(j=0;j<col;j++)
        {
                cout<<b[i][j]<<"\t";
        }
        cout<<"||"<<endl;
}
//substraction OF MATRIX A & B    .
for(i=0;i<row;i++)
{
        for(j=0;j<col;j++)
        {
                c[i][j]=a[i][j]-b[i][j];
        }

}
cout<<endl<<endl<<"YOUR MATRIX SUBSTRACTION IS :"<<endl;
for(i=0;i<row;i++)
{    cout<<"||"<<"\t";
        for(j=0;j<col;j++)
        {
                cout<<c[i][j]<<"\t";
        }
        cout<<"||"<<endl;
}
break;

case 3: //MULTIPLICATION of matrix
        // matrix a
        cout<<"ENTER MATRIX 'A' "<<endl;
        cout<<"ENTER NO OF ROWS OF MATRIX A"<<endl;
```

```
cin>>row;
cout<<"ENTER NO OF COLOUMNS OF MATRIX A"<<endl;
cin>>col;
cout<<"ENTER MATRIX ELEMENTS :"<<endl;
for(i=0;i<row;i++)
{
        for(j=0;j<col;j++)
        {
                cout<<"\t"<<"a["<<i<<"]["<<j<<"]\t";
                cin>>a[i][j];
        }

}
cout<<endl<<endl<<"YOUR MATRIX A IS :"<<endl;
for(i=0;i<row;i++)
{   cout<<"||"<<"\t";
        for(j=0;j<col;j++)
        {
                cout<<a[i][j]<<"\t";
        }
        cout<<"||"<<endl;
}
//matrix b
cout<<"ENTER MATRIX 'B' "<<endl;
cout<<"ENTER NO OF ROWS OF MATRIX B"<<endl;
cin>>row;
cout<<"ENTER NO OF COLOUMNS OF MATRIX B"<<endl;
cin>>col;
cout<<"ENTER MATRIX ELEMENTS :"<<endl;
for(i=0;i<row;i++)
{
        for(j=0;j<col;j++)
        {
                cout<<"\t"<<"b["<<i<<"]["<<j<<"]\t";
                cin>>b[i][j];
        }

}
cout<<endl<<endl<<"YOUR MATRIX B IS :"<<endl;
for(i=0;i<row;i++)
{   cout<<"||"<<"\t";
        for(j=0;j<col;j++)
        {
                cout<<b[i][j]<<"\t";
        }
        cout<<"||"<<endl;
}
//MULTIPLICATION OF MATRIX A & B .
for(i=0;i<row;i++)
{
        for(j=0;j<col;j++)
        {
          for(k=0;k<col;k++)
          {
                c[i][j]=a[i][k]*b[k][j]+c[i][j];
          }
        }
```

```
                }
                cout<<endl<<endl<<"YOUR MATRIX multiplication IS :"<<endl;
                for(i=0;i<row;i++)
                {    cout<<"||"<<"\t";
                        for(j=0;j<col;j++)
                        {
                                cout<<c[i][j]<<"\t";
                        }
                        cout<<"||"<<endl;
                }
                break;

case 4: //transpose of matrix
                // matrix a
                do{
                cout<<"ENTER MATRIX 'A' "<<endl;
                cout<<"ENTER NO OF ROWS OF MATRIX A"<<endl;
                cin>>row;
                cout<<"ENTER NO OF COLOUMNS OF MATRIX A"<<endl;
                cin>>col;
                cout<<endl<<endl;
                }while(row!=col);
                cout<<"ENTER MATRIX ELEMENTS :"<<endl;
                for(i=0;i<row;i++)
                {
                        for(j=0;j<col;j++)
                        {
                                cout<<"\t"<<"a["<<i<<"]["<<j<<"]\t";
                                cin>>a[i][j];
                        }

                }
                cout<<endl<<endl<<"YOUR MATRIX A IS :"<<endl;
                for(i=0;i<row;i++)
                {    cout<<"||"<<"\t";
                        for(j=0;j<col;j++)
                        {
                                cout<<a[i][j]<<"\t";
                        }
                        cout<<"||"<<endl;
                }

                //transpose of matrix
                cout<<"TRANSPOSE OF MATRIX IS :"<<endl<<endl;
                for(i=0;i<row;i++)
                {    cout<<"||"<<"\t";
                        for(j=0;j<col;j++)
                        {
                                cout<<a[j][i]<<"\t";
                        }
                        cout<<"||"<<endl;
                }
                break;

case 5: //SUMMAtion of diagonal elements of matrix
                // matrix a
                cout<<"ENTER MATRIX 'A' "<<endl;
```

```
                cout<<"ENTER NO OF ROWS OF MATRIX A"<<endl;
                cin>>row;
                cout<<"ENTER NO OF COLOUMNS OF MATRIX A"<<endl;
                cin>>col;
                cout<<"ENTER MATRIX ELEMENTS :"<<endl;
                for(i=0;i<row;i++)
                {
                        for(j=0;j<col;j++)
                        {
                                cout<<"\t"<<"a["<<i<<"]["<<j<<"]\t";
                                cin>>a[i][j];
                        }

                }
                cout<<endl<<endl<<"YOUR MATRIX A IS :"<<endl;
                for(i=0;i<row;i++)
                {    cout<<"||"<<"\t";
                        for(j=0;j<col;j++)
                        {
                                cout<<a[i][j]<<"\t";
                        }
                        cout<<"||"<<endl;
                }
                for(i=0;i<row;i++)
                {
                        for(j=0;j<col;j++)
                        {
                                if(i==j)
                                {
                                  m=a[i][j]+m;
                                }
                        }

                }

                //SUMMATION OF diagonal elements of matrix
                cout<<"SUMMATION OF DIAGONAL ELEMENTS OF MATRIX IS
:"<<endl<<endl;
                cout<<m<<endl<<endl;
                break;


        case 6: //upper triangular matrix matrix
                // matrix a
                cout<<"ENTER MATRIX 'A' "<<endl;
                cout<<"ENTER NO OF ROWS OF MATRIX A"<<endl;
                cin>>row;
                cout<<"ENTER NO OF COLOUMNS OF MATRIX A"<<endl;
                cin>>col;
                cout<<"ENTER MATRIX ELEMENTS :"<<endl;
                for(i=0;i<row;i++)
                {
                        for(j=0;j<col;j++)
                        {
                                cout<<"\t"<<"a["<<i<<"]["<<j<<"]\t";
                                cin>>a[i][j];
```

```
                }

        }
        cout<<endl<<endl<<"YOUR MATRIX A IS :"<<endl;
        for(i=0;i<row;i++)
        {    cout<<"||"<<"\t";
                for(j=0;j<col;j++)
                {
                        cout<<a[i][j]<<"\t";
                }
                cout<<"||"<<endl;
        }
        for(i=0;i<row;i++)
        {
                for(j=0;j<col;j++)
                {
                        if(i==j)
                        {
                          c[p][q]=a[i][j];
                          p++;
                          q++;
                        }
                }

        }
        for(p=0;p<row-1;p++)
        {
                for(q=0;q<col-1;q++)
                {
                        if(c[p][q]==0)
                        {
                          c[p][q]=a[i][j];
                          p++;
                          q++;
                        }
                }

        }


                default : break;

        }       cout<<"DO YOU WANT TO CONTINUE ..(Y_es/N_o) :"<<endl<<endl;
                cin>>chr;
    }while(chr=='y'||chr=='Y');

}
```

Practical No : 04(A)

Practical Title: Write a C++ program to perform String operations.

Aim: Write C++ program for string operations- copy, concatenate, check substring, equal, reverse and length.

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 operation (explain each operation in detail with example)

- Copy string
- Concatenate string
- Equal string
- Length of string
- Substring
- Reverse string

Algorithms :

1. Start
2. Accept two string from user i.e. str1 and str2
3. Write function to copy str2 to str1
4. Perform concatenation of two string
5. Write function to find length of string
6. Function to check given string are equal or not
7. Function to find subtring
8. Write function to reverse string
9. Display all string operation
10. End

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.

Program :-

```
#include<iostream>
#include<string.h>
using namespace std;
int main()
{
 int i, j; // INDEXING VARIABLES.
 int choice; // VARIABLE USED TO SELECT CASE FROM SWITCH CASE.
 int len1=0, len2=0; //VARIABLES TO STORE LENGTHS OF STRING.
 char str_1[100], str_2[100], cpy[100], concn[200], revers[100]; //VARIABLS TO STORE THE STRINGS .
 char prob; // VARIABLE USED TO CONTINUE SWITCH CASE.
 int count1 = 0, count2 = 0, flag;  // FOR INITIALIZING THE VALUES OF 'IF' OR'IF ELSE' STATEMENTS.
 do{
 do{
 // DRIVEN MENU.
 cout<<"ENTER THE ACTION NO YOU WANT TO PERFORM:"<<endl<<endl<<endl;
 cout<<"**********************"<<endl<<endl;
 cout<<"***MENU***"<<endl;
 cout<<"1).STRING LENGTH"<<endl;
 cout<<"2).STRING COPY"<<endl;
 cout<<"3).STRING COMPARE"<<endl;
 cout<<"4).STRING CONCANATE"<<endl;
 cout<<"5).STRING REVERSE"<<endl;
 cout<<"6).SUB-STRING "<<endl;
 cout<<"7).EXIT"<<endl<<endl;
 cout<<"**********************"<<endl<<endl;

 // SELECTION CASE.
 cin>>choice;
   if(choice<1||choice>7)
   {
     cout<<"ENTER A VALID INPUT\n\n";
   }
 }while(choice<1||choice>7);


 switch(choice)
 {
   case 1:   //STRING LENGTH.

           // ACCEPT THE STRING.
           cout<<"enter the string :"<<endl<<endl;
           cin>>str_1;
           // CALCULATE STRING LENGTH.
           for(i=0;str_1[i]!='\0';i++)
              {
                 len1=len1+1;
```

```
        }

        // PRINT OUTPUT  .
        cout<<"the length of string is ::"<<len1<<endl;
        break;

case 2:    //COPYING STRING.

        // ACCEPT THE STRING.
        cout<<"enter the first string"<<endl;
        cin>>str_1;
        cout<<endl<<endl;
        // COPY OPERATION.
        cout<<"copiying in process....."<<endl<<endl;
        for(i=0;str_1[i]!='\0';i++)
         {
            str_2[i]=str_1[i];
         }

        // PRINT OUTPUT.
        cout<<"the copied second string ::"<<str_2<<endl<<endl;
        break;

case 3:    //CHECK FOR IDENTICAL STRINGS.

        //ACCEPT THE TWO STRINGS.
        cout<<"enter the first string :"<<endl;
                    cin>>str_1;
        cout<<"enter the second string :"<<endl;
        cin>>str_2;

        // CALCULATING LENGTH OF BOTH STRINGS.
        len1=strlen(str_1);
        len2=strlen(str_2);

        // COMPARING TWO STRINGS...
        if(len1==len2)
        {
        for(i=1;str_1[i]!='\0';i++)
        {
           if(str_1[i]==str_2[i])
             flag=0;
           else
             flag=1;
        }
        }
        else
          flag==1;

        // PRINT OUTPUT.
        if(flag==0)
           cout<<"the strings are identical"<<endl<<endl;
        else
           cout<<"the strings are not identical"<<endl<<endl;
        break;


case 4:   // CODE FOR CONCANATING STRINGS.
        //ACCEPT THE TWO STRINGS.
        cout<<"enter the first string :"<<endl;
                    cin>>str_1;
        cout<<"enter the second string :"<<endl;
```

```
        cin>>str_2;

        //MERGING OF TWO STRINGS.
        cout<<"the concaniated string is : "<<endl;
        strcat(str_1,str_2);

        // PRINT OUTPUT.
        cout<<str_1<<endl;
        break;

case 5 : // CODE FOR REVERSING STRING.

       // ACCEPT AND CALCULATE THE STRING LENGTH.
       cout<<"enter the string"<<endl<<endl;
             cin>>str_1;
        len1=strlen(str_1);

        //OPERATION FOR REVERSING THE STRING.
        for(i=0;i<len1;i++)
        {
              str_2[i]=str_1[len1-(i+1)];
        }

        // PRINT OUTPUT.
             cout<<endl<<endl<<"the reverse string is:\t";
       for(i=0;i<len1;i++)
        {
                  cout<<str_2[i];
       }
             cout<<endl<<endl;
       break;

 case 6:    // CODE FOR SUB-STRING.

          //ACCEPT THE TWO STRINGS.
          cout<<"enter the main string :"<<endl;
                       cin>>str_1;
          cout<<"enter a string :"<<endl;
          cin>>str_2;

          //OPERATION FOR CHECKING THE SUB STRING.
          while (str_1[count1] != '\0')
          count1++;
          while (str_2[count2] != '\0')
          count2++;
          for (i = 0; i <= count1 - count2; i++)
          {
          for (j = i; j < i + count2; j++)
          {
                flag = 1;
                if (str_1[j] != str_2[j - i])
                {
                  flag = 0;
                  break;
                }
          }
          if (flag == 1)
          break;
          }

          // PRINT OUTPUT.
          if (flag == 1)
```

```
            cout<<"IT IS A SUB STRING OF MAIN STRING\n\n";
          else
            cout<<"IT IS NOT A SUB STRING OF MAIN STRING\n\n";
          break;


   default : break;
  }
   // TO CONTINUE THE PROGRAM
   cout<<"do you want to continue (yes/ no).....";
   cin>>prob;

  }while(prob=='y'||prob=='Y');

}
```

/* output

```
sl280@sl280-HP-dx2480-MT-VP562PA:~$ g++ strsw.cpp
sl280@sl280-HP-dx2480-MT-VP562PA:~$ ./a.out
ENTER THE ACTION NO YOU WANT TO PERFORM:


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

***MENU***
1).STRING LENGTH
2).STRING COPY
3).STRING COMPARE
4).STRING CONCANATE
5).STRING REVERSE
6).SUB-STRING
7).EXIT

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

1
enter the string :

pvgcoe
the length of string is ::6
do you want to continue (yes/ no).....y
ENTER THE ACTION NO YOU WANT TO PERFORM:


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

***MENU***
1).STRING LENGTH
2).STRING COPY
3).STRING COMPARE
4).STRING CONCANATE
5).STRING REVERSE
6).SUB-STRING
7).EXIT

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

2
enter the first string
```

pvgcoe

copiying in process.....

the copied second string ::pvgcoe

do you want to continue (yes/ no).....y
ENTER THE ACTION NO YOU WANT TO PERFORM:

****

MENU
1).STRING LENGTH
2).STRING COPY
3).STRING COMPARE
4).STRING CONCANATE
5).STRING REVERSE
6).SUB-STRING
7).EXIT

****

3
enter the first string :
pvgcoe
enter the second string :
pvgcoe
the strings are identical

do you want to continue (yes/ no).....y
ENTER THE ACTION NO YOU WANT TO PERFORM:

****

MENU
1).STRING LENGTH
2).STRING COPY
3).STRING COMPARE
4).STRING CONCANATE
5).STRING REVERSE
6).SUB-STRING
7).EXIT

****

3
enter the first string :
pvgcoe
enter the second string :
pvgcoen
the strings are not identical

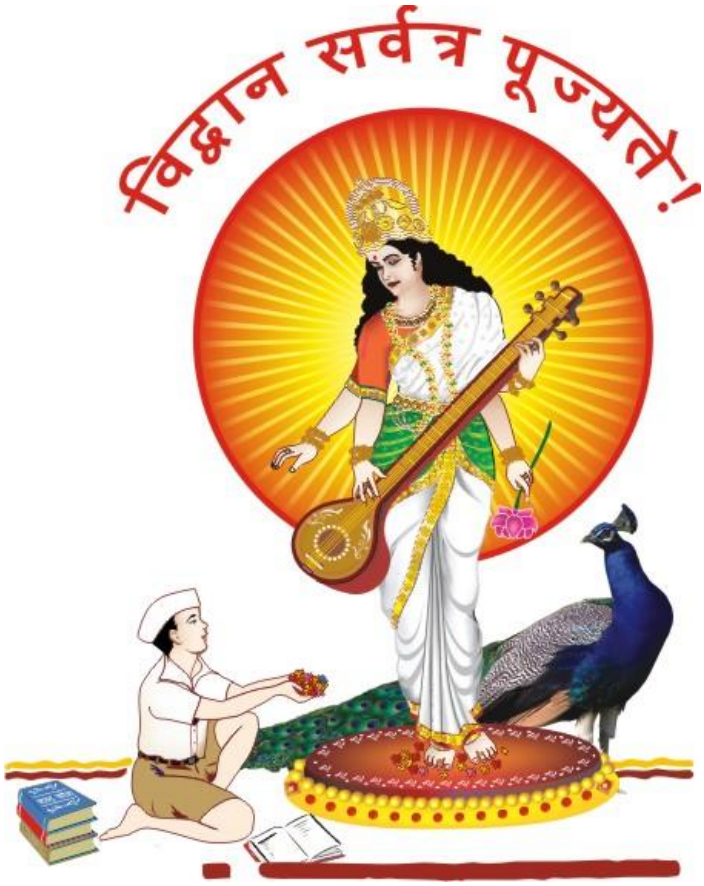
do you want to continue (yes/ no).....y
ENTER THE ACTION NO YOU WANT TO PERFORM:

****

```
***MENU***
1).STRING LENGTH
2).STRING COPY
3).STRING COMPARE
4).STRING CONCANATE
5).STRING REVERSE
6).SUB-STRING
7).EXIT

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

4
enter the first string :
pvgcoe
enter the second string :
_nashik
the concaniated string is :
pvgcoe_nashik
do you want to continue (yes/ no).....y
ENTER THE ACTION NO YOU WANT TO PERFORM:


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

***MENU***
1).STRING LENGTH
2).STRING COPY
3).STRING COMPARE
4).STRING CONCANATE
5).STRING REVERSE
6).SUB-STRING
7).EXIT

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

5
enter the string

pvgcoe


the reverse string is:  eocgvp

do you want to continue (yes/ no).....y
ENTER THE ACTION NO YOU WANT TO PERFORM:


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

***MENU***
1).STRING LENGTH
2).STRING COPY
3).STRING COMPARE
4).STRING CONCANATE
5).STRING REVERSE
6).SUB-STRING
7).EXIT

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

3
enter the first string :
```

```
pvg
enter the second string :
pvgcoe
the strings are not identical

do you want to continue (yes/ no).....Y


ENTER THE ACTION NO YOU WANT TO PERFORM:


**********

***MENU***
1).STRING LENGTH
2).STRING COPY
3).STRING COMPARE
4).STRING CONCANATE
5).STRING REVERSE
6).SUB-STRING
7).EXIT

**********

6
enter the main string :
PVGCOE
enter a string :
PVG
IT IS A SUB STRING OF MAIN STRING

do you want to continue (yes/ no).....Y
ENTER THE ACTION NO YOU WANT TO PERFORM:


**********

***MENU***
1).STRING LENGTH
2).STRING COPY
3).STRING COMPARE
4).STRING CONCANATE
5).STRING REVERSE
6).SUB-STRING
7).EXIT

**********

6
enter the main string :
PVGCOE
enter a string :
GVC
IT IS NOT A SUB STRING OF MAIN STRING

do you want to continue (yes/ no).....N
sl280@sl280-HP-dx2480-MT-VP562PA:~$
```

GROUP - B



GROUP - B

Practical No : 05(B)

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:

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?

Program :

/*Code for Program to maintain club member‘s information using singly linked list.*/

```
#include<iostream>
using namespace std;
typedef struct stud
{
char PRN[20];
char name[20];
}stud;


typedef struct node
{
stud s;
struct node *next;
}node;


node *createnode()
{
int i,n,cnt=0;
node *p,*head,*t,*st;
head=NULL;
cout<<"\n Enter the no of nodes:";
cin>>n;
for(i=0;i<n-1;i++)
{
p=new node;
if (head==NULL)
{
cout<<"\n Enter the data for President";
cout<<"\n\n Enter PRN number:";
cin>>p->s.PRN;
cout<<"\n Enter the name:";
cin>>p->s.name;
p->next=NULL;
head=p;
```

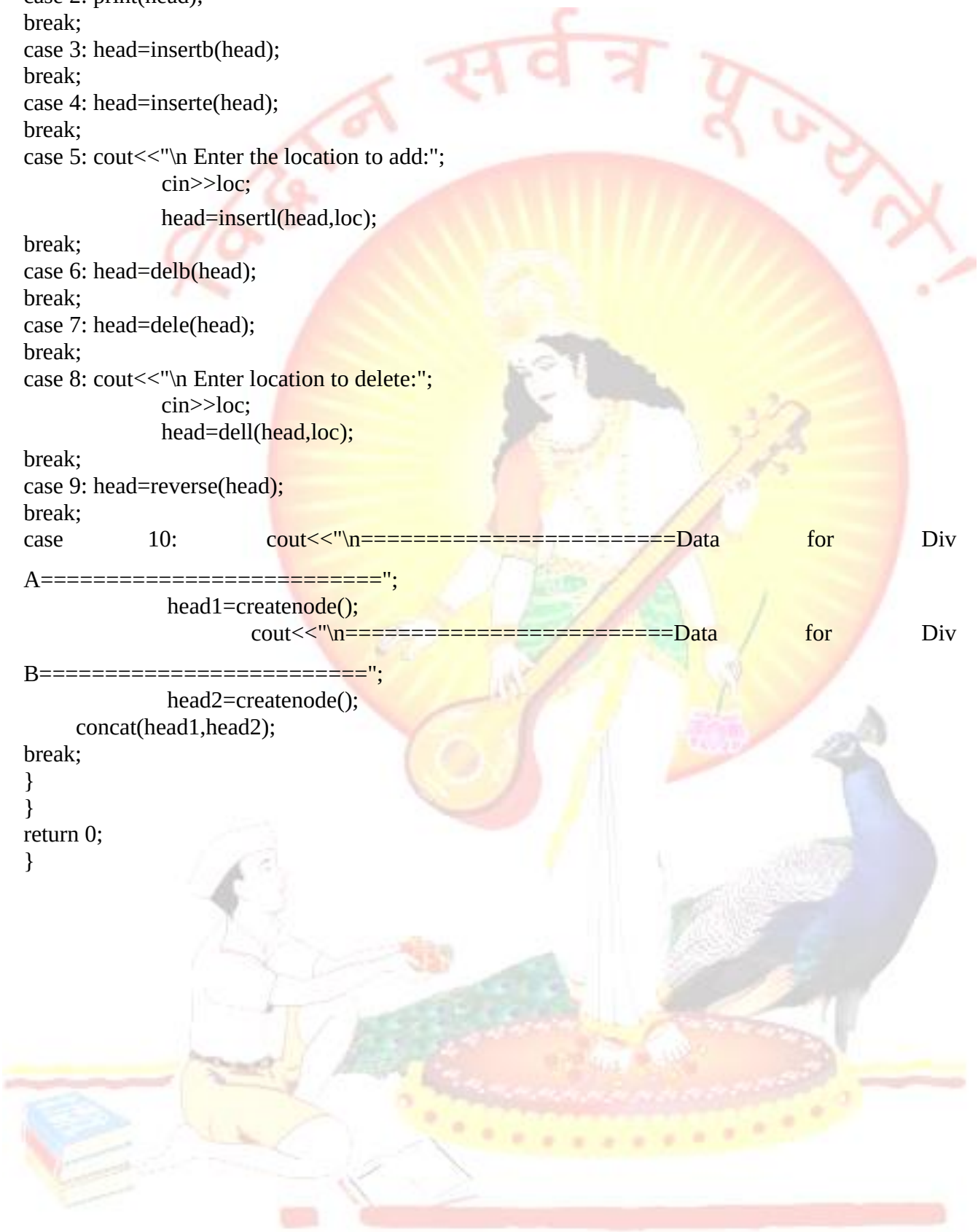
```
cnt++;
}
else
{
cout<<"\n Enter the data for members";
cout<<"\n\n Enter PRN number:";
cin>>p->s.PRN;
cout<<"\n Enter the name:";
cin>>p->s.name;
t=head;
while(t->next!=NULL)
t=t->next;
t->next=p;
cnt++;
}
}

st=new node;
cout<<"\n Enter the data for Secretary";
cout<<"\n\n Enter PRN number:";
cin>>st->s.PRN;
cout<<"\n Enter the name:";
cin>>st->s.name;
t=head;
while(t->next!=NULL)
t=t->next;
t->next=st;
return head;
}


void print(node *head)
{
node *p;
int cnt=0;
p=head;
cout<<"\n ======Club data=====";
while(p!=NULL)
{
cout<<"\n PRN number: "<<p->s.PRN<<"\n Name: "<<p->s.name<<"====>";
p=p->next;
cnt++;
}
cout<<"\n Total no of nodes="<<cnt;
}

node *insertb(node *head)
{
node *p;
p=new node;
cout<<"\n Enter the data of student";
cout<<"\n\n Enter PRN number:";
cin>>p->s.PRN;
cout<<"\n Enter the name:";
cin>>p->s.name;
if(head==NULL)
{
```

```
head=p;
return p;
}
p->next=head;
head=p;
return head;
}


node *insertl(node *head,int loc)
{
node *p,*q;
p=new node;
cout<<"\n Enter the data of student";
cout<<"\n\n Enter PRN number:";
cin>>p->s.PRN;
cout<<"\n Enter the name:";
cin>>p->s.name;
if(head==NULL)
{
head=p;
return p;

}
for(q=head;q->next!=NULL;q=q->next);
q->next=p;
return head;
}
//node *insertl(node *head,int loc)
//{

node *inserte(node *head)
{
node *p,*q;
int i,loc;
p=new node;
cout<<"\n Enter the data of student";
cout<<"\n\n Enter PRN number:";
cin>>p->s.PRN;
cout<<"\n Enter the name:";
cin>>p->s.name;
p->next=NULL;
if(loc==1)
{
p->next=head;
head=p;
return head;
}
q=head;
for(i=1;i<loc-1;i++)
if(q!=NULL)
q=q->next;
else
{
cout<<"\n Over flow";
return head;
}
```

```
p->next=q->next;
q->next=p;
return head;
}

node *delb(node *head)
{
node *p;
if(head==NULL)
{
cout<<"\n List is empty";
return head;
}
p=head;
head=head->next;
p->next=NULL;
delete p;
return head;
}

node *dele(node *head)
{
node *p,*q;
if(head==NULL)
{
cout<<"\n list is empty";
return head;
}
for(q=head;q->next->next!=NULL;q=q->next);
p=q->next;
q->next=NULL;
delete p;
return head;
}

node *dell(node *head,int loc)

{
node *p,*q;
int i;
if (head==NULL)
{
cout<<"list is empty";
return head;
}
if(loc==1)
{
p=head;
head=head->next;
p->next=NULL;
delete p;
return head;
}
q=head;
for(i=1;i<loc-1;i++)
if(q->next!=NULL)
q=q->next;
```

```
else
{
cout<<"Underflow";
return head;
}
p=q->next;
q->next=p->next;
p->next=NULL;
delete p;
return head;
}

node *reverse(node *head)
{
node *p,*q,*r;
p=NULL;
q=head;
r=q->next;
while(q!=NULL)
{
q->next=p;
p=q;
q=r;
if(q!=NULL)
r=q->next;
}

return p;
}
void concat(node *head1,node *head2)
{
node *p,*q;
p=head1;
q=head2;
while(p->next!=NULL)
p=p->next;
p->next=head2;
print(head1);
}

int main()
{
node *head, *head1, *head2;
head= NULL;
int ch,ele,loc;
head=NULL;
head1=head2=NULL;
while(ch!=11)
{

cout<<"\n ========== pinacle club ======= \n ";
cout<<"\n 1.Create \n 2.Print \n 3.Insert New President \n 4.Insert New Secretary \n 5.Insert

Member";
cout<<"\n 6.Delete President \n 7.Delete Secretary \n 8.Delete member \n 9.Reverse \n

10.Concatenate \n 11.Exit";
cout<<"\n Enter ur choice:";
```

```
cin>>ch;
switch(ch)
{
case 1: head=createnode();
break;
case 2: print(head);
break;
case 3: head=insertb(head);
break;
case 4: head=inserte(head);
break;
case 5: cout<<"\n Enter the location to add:";
                cin>>loc;
                head=insertl(head,loc);
break;
case 6: head=delb(head);
break;
case 7: head=dele(head);
break;
case 8: cout<<"\n Enter location to delete:";
                cin>>loc;
                head=dell(head,loc);
break;
case 9: head=reverse(head);
break;
case 10: cout<<"\n=======================Data for Div A=========================";
                head1=createnode();
                        cout<<"\n=======================Data for Div B=========================";
                head2=createnode();
        concat(head1,head2);
break;
}
}
return 0;
}
```



Practical No: 06(B)

Practical Title: Write C++ program for storing binary number using 8/8 doubly linked lists.

Aim: Write C++ program for storing binary number using doubly linked lists. Write functions- a) to compute 1‘s and 2‘s complement b) add two binary numbers

Pre-requisite:

- Knowledge of Doubly Linked List
- Representation of binary number
- Knowledge of finding 1's and 2's complement of a number.

Objective:

- To store binary numbers using doubly linked list.
- To find 1's and 2's complement of a binary number.
- To add two binary numbers

Input:

Two Binary numbers

Outcome:

- 1's and 2's complement of a binary number.
- Result of addition of two binary numbers.

Theory :

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

- **Doubly 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 store binary number using 8/8 doubly linked lists.

A	P	J	Total	Dated Sign
3	4	3	10	

Questions:

1. What is a binary number?
2. How to represent binary number using doubly linked list?
3. How to find 1's and 2's complement of a binary number?
3. What is doubly linked list?
4. How to insert and delete elements from doubly linked list?

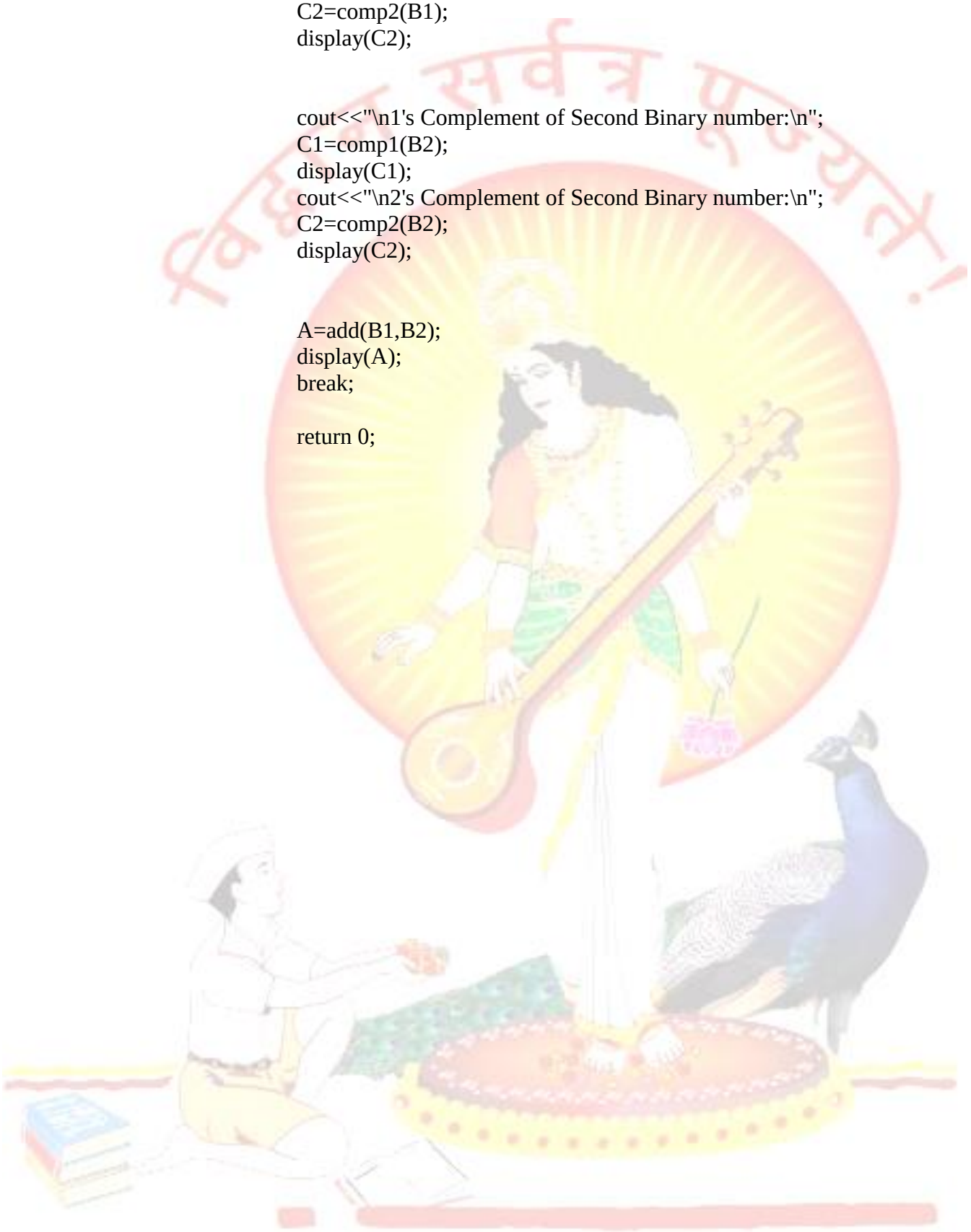
Program :

```
#include<iostream.h>
#include<stdlib.h>
//using namespace std;
struct node *create();
void display(struct node *);
struct node *reverse(struct node *);
struct node *comp1(struct node *);
struct node *comp2(struct node *);
struct node * add(struct node *, struct node *);
struct node
{
        int data;
        struct node *next;
        struct node *prev;
}*B1=NULL,*B2=NULL,*C1=NULL,*C2=NULL,*A=NULL;
int main()
{
        int ch;
        cout<<"Enter First Binary no.:\n";
        B1=create();
        cout<<"\n First Binary no. \n";
        display(B1);
        cout<<"\nEnter the Second Binary no.:\n";
        B2=create();
        cout<<"\n Second Binary no. \n";
        display(B2);
        while(1)
                {
                cout<<"\n\nMenu:";
                cout<<"\n1.1's Complement and 2's complement of First Binary number.";
                cout<<"\n2.1's Complement and 2's complement of Second Binary
                number."; cout<<"\n3.Addition of two binary nos."; cout<<"\n4.Exit.";

                cout<<"\n\nEnter choice:";
                cin>>ch;
                switch(ch)
                        {
                        case 1:
```

break; case 3:

```
                        break;
                        case 2:
```

```
case 4: cout<<"\n1's Complement of First Binary number:\n";
        C1=comp1(B1);
        display(C1);
        cout<<"\n2's Complement of First Binary number:\n";
        C2=comp2(B1);
        display(C2);


        cout<<"\n1's Complement of Second Binary number:\n";
        C1=comp1(B2);
        display(C1);
        cout<<"\n2's Complement of Second Binary number:\n";
        C2=comp2(B2);
        display(C2);


        A=add(B1,B2);
        display(A);
        break;

        return 0;
```

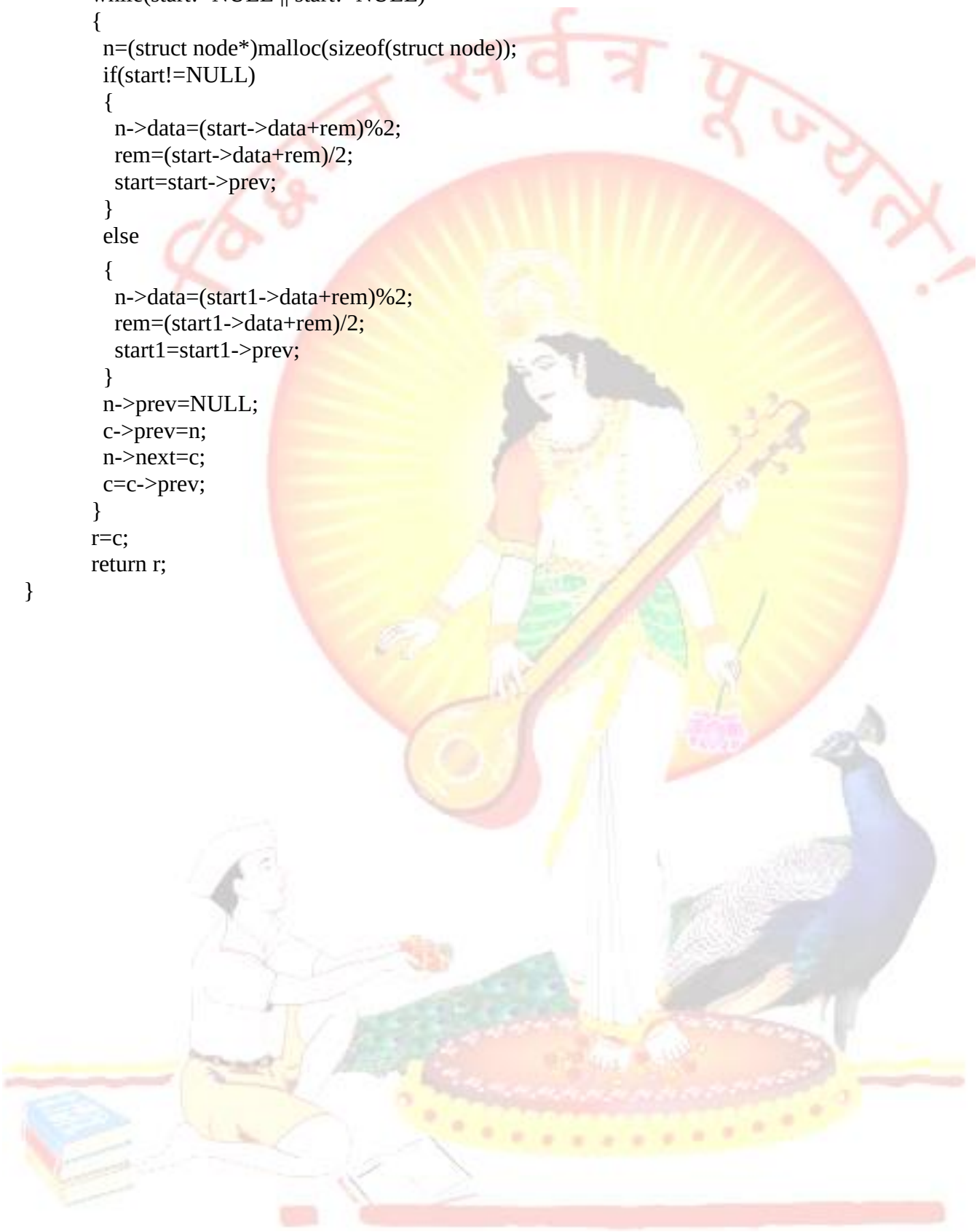


```
                        default:
                                cout<<"Invalid Choice.";
                        break;
                        }
                }
}
struct node *create()

{
        char ch;
        struct node *new_node,*current,*start=NULL;
        do
        {
        new_node=(struct node*)malloc(sizeof(struct node));

                label:
                cout<<"\nEnter Binary digit: ";
                cin>>new_node->data;
                new_node->next=NULL;
                new_node->prev=NULL;
                if(new_node->data!=0 && new_node->data!=1)
                        {
                        cout<<"\nEnter Valid binary digit.\n\n";
                        goto label;
                        }
                else
                        {
                        if(start==NULL)

                                {

                                start=new_node;

                                current=new_node;

                                }

                        else


                                {

                                current->next=new_node;

                                new_node->prev=current;
                                current=new_node;
```

```
                        }
                }

                cout<<"\nDo you want to add more digits?(y/n):";

                cin>>ch;

        }while(ch!='n');

        return start;
}


void display(struct node *new_node)
{

        while(new_node!=NULL)

                {

                cout<<new_node->data<<"\t";

                new_node=new_node->next;

                }

        cout<<"\n";

}
struct node *comp1(struct node *new_node)
{
        struct node *n_node,*current,*start=NULL;

        while(new_node!=NULL)
        {

                n_node=(struct node*)malloc(sizeof(struct
                node)); if(new_node->data==1)
                        {
                        n_node->data=0;
                        }
                else
                        {
                        n_node->data=1;
                        }
```

```
                n_node->next=NULL;
                n_node->prev=NULL;
                if(start==NULL)

                        {

                        start=n_node;

                        current=n_node;

                        }

                else

                        {

                        current->next=n_node;

                        n_node->prev=current;
                        current=n_node;
                        }
        new_node=new_node->next;
        }
        return start;
}
struct node *comp2(struct node *temp)
{
        struct node *n1,*c,*start=NULL;
        c=comp1(temp);
        n1=(struct node*)malloc(sizeof(struct node));
        n1->data=1;
        n1->next=NULL;
        n1->prev=NULL;
        start=add(c,n1);
        return start;
}
struct node * add(struct node *start, struct node *start1)
{
        struct node *n, *c,*r=NULL;
        int rem=0;
        while(start->next!=NULL)
        start=start->next;
        while(start1->next!=NULL)
        start1=start1->next;
        while(start!=NULL && start1!=NULL)
        {
         n=(struct node*)malloc(sizeof(struct node));
         n->data=(start->data+start1->data+rem)%2;
         rem=(start->data+start1->data+rem)/2;
         n->next=NULL;
         n->prev=NULL;
         if(r==NULL)
         r=c=n;
         else
         {
          c->prev=n;
          n->next=c;
```

```
            c=c->prev;
           }
           start=start->prev;
           start1=start1->prev;
          }
          while(start!=NULL || start!=NULL)
          {
           n=(struct node*)malloc(sizeof(struct node));
           if(start!=NULL)
           {
            n->data=(start->data+rem)%2;
            rem=(start->data+rem)/2;
            start=start->prev;
           }
           else
           {
            n->data=(start1->data+rem)%2;
            rem=(start1->data+rem)/2;
            start1=start1->prev;
           }
           n->prev=NULL;
           c->prev=n;
           n->next=c;
           c=c->prev;
          }
          r=c;
          return r;
}
```



Practical No:07(B)

Practical Title: Write C++ program to store set of negative and positive numbers using linked list

Aim: Write C++ program to store set of negative and positive numbers using linked list. Write functions

a) Insert numbers

b) Delete nodes with negative numbers

c) To create two more linked lists using this list, one containing all positive numbers and other containing negative numbers

d) For two lists that are sorted; Merge these two lists into third resultant list that is sort

Pre-requisite:

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

Objective:

- To store set of positive and negative numbers using linked list.

Input:

Number of Elements
Positive and negative numbers

Outcome:

- List of positive numbers
- List of negative numbers
- Result of Sort and merge operations on linked list
-

Theory : write short theory of linked list.

Explain logic/algorithms for positive and negative number

Explain logic/algorithms for sorting and merging linked list.

Algorithms:

(Write your algorithms for your program.)

Flowchart :

(Draw flowchart for above algorithms.)

Conclusion:

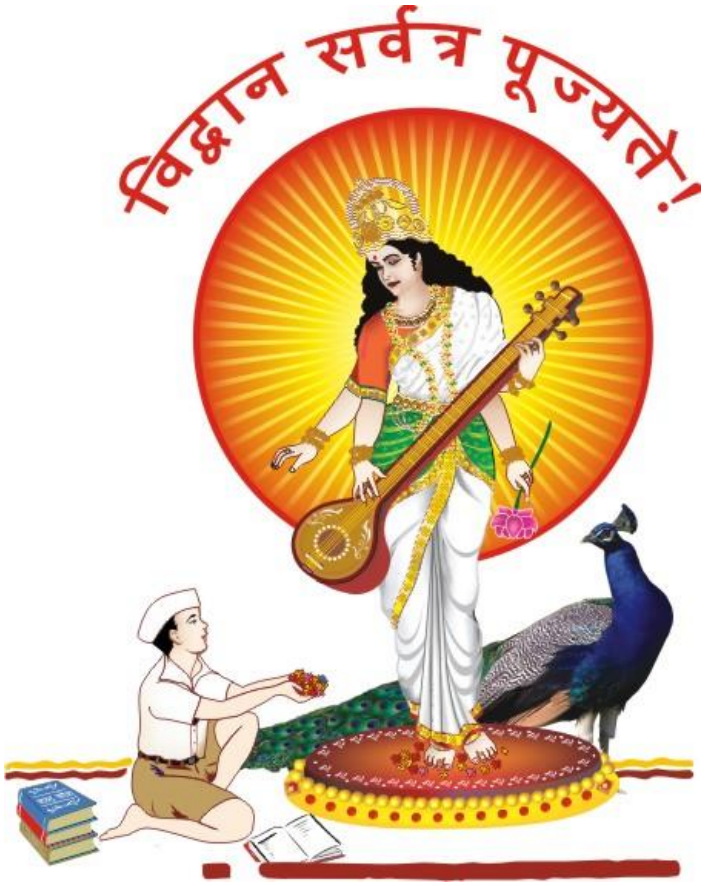
By this way, we can store set of negative and positive numbers using linked list

A	P	J	Total	Dated Sign
3	4	3	10	

Question Bank:

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 - C



GROUP - C

Practical No:08(C)

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?

Program :

```
#include  <iostream>
using namespace std;
#define size 10

class stackexp
{
   int top;
   char stk[size];
public:
   stackexp()
   {
    top=-1;
   }
   void push(char);
   char pop();
   int isfull();
   int isempty();
};

void stackexp::push(char x)
{
   top=top+1;
   stk[top]=x;
}

char stackexp::pop()
{
   char s;
   s=stk[top];
   top=top-1;
   return s;
}

int stackexp::isfull()
{
   if(top==size)
      return 1;
   else
      return 0;
}

int stackexp::isempty()
{
   if(top==-1)
      return 1;
   else
      return 0;
}

int main()
{
   stackexp s1;
   char exp[20],ch;
   int i=0;
```

```
    cout <<"\n\t!!Well Formness of Parenthesis..!!!!"<< endl; // prints !!!Hello World!!!
    cout<<"\nEnter the expression to check whether it is in well form or not : ";
    cin>>exp;
    if((exp[0]==')')||(exp[0]==']')||(exp[0]=='}'))
    {
       cout<<"\n Invalid Expression.....\n";
       return 0;
    }
    else
    {
       while(exp[i]!='\0')
       {
          ch=exp[i];
          switch(ch)
          {
          case '(':s1.push(ch);break;
          case '[':s1.push(ch);break;
          case '{':s1.push(ch);break;
          case ')':s1.pop();break;
          case ']':s1.pop();break;
          case '}':s1.pop();break;
          }
          i=i+1;
       }
    }
    if(s1.isempty())
    {
       cout<<"\nExpression is well parenthesis...\n";
    }
    else
    {
       cout<<"\nSorry !!! Invalid Expression or not in well parenthesized....\n";
    }
    return 0;
}
```

/****************** output **

!!Well Formness of Parenthesis..!!!!

Enter the expression to check whether it is in well form or not : (m<(n[8]+0))

Expression is well parenthesis...

!!Well Formness of Parenthesis..!!!!

Enter the expression to check whether it is in well form or not :)(m+n)*(a-b)

Invalid Expression.....

!!Well Formness of Parenthesis..!!!!

Enter the expression to check whether it is in well form or not : (m+b(n[8)/2+3)

Sorry !!! Invalid Expression or not in well parenthesized....

Practical No:09(C)

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

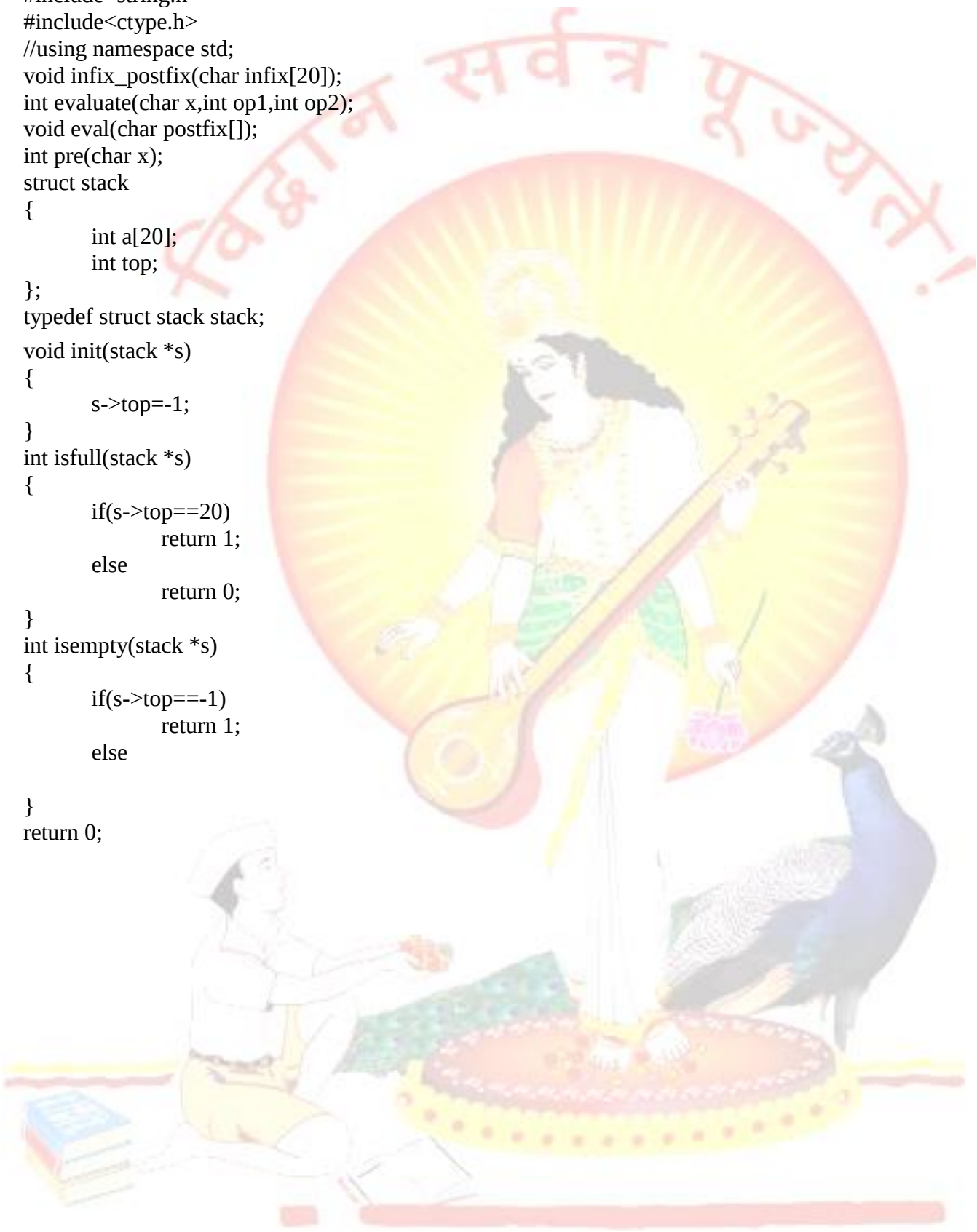
A	P	J	Total	Dated Sign
3	4	3	10	

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

Program :

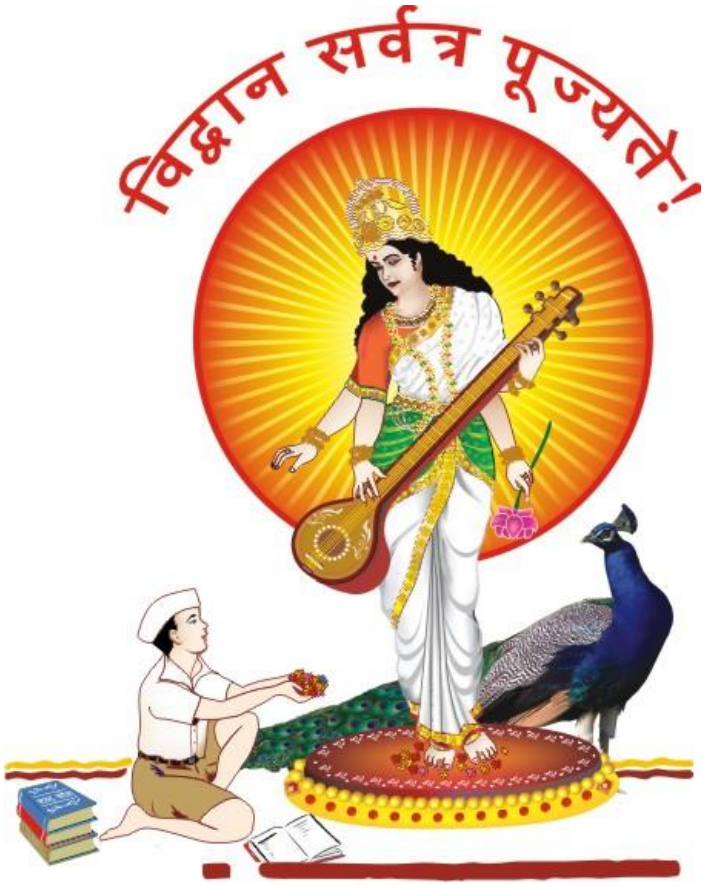
```
#include<iostream.h>
#include<conio.h>
#include<string.h>
#include<ctype.h>
//using namespace std;
void infix_postfix(char infix[20]);
int evaluate(char x,int op1,int op2);
void eval(char postfix[]);
int pre(char x);
struct stack
{
        int a[20];
        int top;
};
typedef struct stack stack;
void init(stack *s)
{
        s->top=-1;
}
int isfull(stack *s)
{
        if(s->top==20)
                return 1;
        else
                return 0;
}
int isempty(stack *s)
{
        if(s->top==-1)
                return 1;
        else

}
return 0;
```



```
void push(stack *s,char x)
{
        s->top=s->top+1;
        s->a[s->top]=x;
}
int pop(stack *s)
{
        char x;
        x=s->a[s->top];
        s->top=s->top-1;
        return x;
}
void main()
{
        clrscr();
        char infix[20];
        cout<<"Enter infix expression:";
        cin>>infix;
        infix_postfix(infix);
        getch();
}
void infix_postfix(char infix[20])
{
        stack *s;
        char postfix[20],x,y;
        int i,j=0;
        init(s);
        for(i=0;infix[i]!='\0';i++)
                {
                y=infix[i];
                if((y>='a' && y<='z') || (y>='A' && y<='Z'))
                        {
                        postfix[j++]=y;
                        }
                else if(y=='(')
                        {
                        push(s,y);
                        }
                else if(y==')')
                        {
                        while((x=pop(s))!='(')
                        postfix[j++]=x;
                        }
                else
                        {
                        while(pre(y)<=pre(s->a[s->top])&&!isempty(s))
                                {
                                postfix[j++]=pop(s);
                                }
                        push(s,y);
                        }
                }
```

```
                while(!isempty(s))
                    {
                        postfix[j++]=pop(s);
                    }

}
int pre(char x)
{
postfix[j]='\0'; cout<<postfix; eval(postfix);
if(x=='(') return 0;
        else if(x=='+' || x=='-')
                return 1;
        else if(x=='/' || x=='*' || x=='%')
                return 2;
        else
        return 3;
}
void eval(char postfix[])
 {
  stack s;
  char x;
  int op1,op2,val,i;
  init(&s);
  for(i=0;postfix[i]!='\0';i++)
   {
    x=postfix[i];
    if(isalpha(x))
     {
      cout<<"\nEnter the value of:"<<x;
      cin>>val;
      push(&s,val);
     }
  else
  {
    op2=pop(&s);
    op1=pop(&s);
    val=evaluate(x,op1,op2);
    push(&s,val);
    }
  }
 val=pop(&s);
 cout<<"\nValue of expression:"<<val;
}


 int evaluate(char x,int op1,int op2)
 {
  switch(x)
  {
   case '+': return(op1+op2);
   case '-': return(op1-op2);
   case '*': return(op1*op2);
   case '/': return(op1/op2);
   case '%': return(op1%op2);
  }
 }
```

GROUP - D



GROUP - D

Practical No:10(D)

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 :
Write your own algorithms

Flowchart :
Draw flowchart for above algorithms

Conclusion:

By this way, we can perform different operations on queue

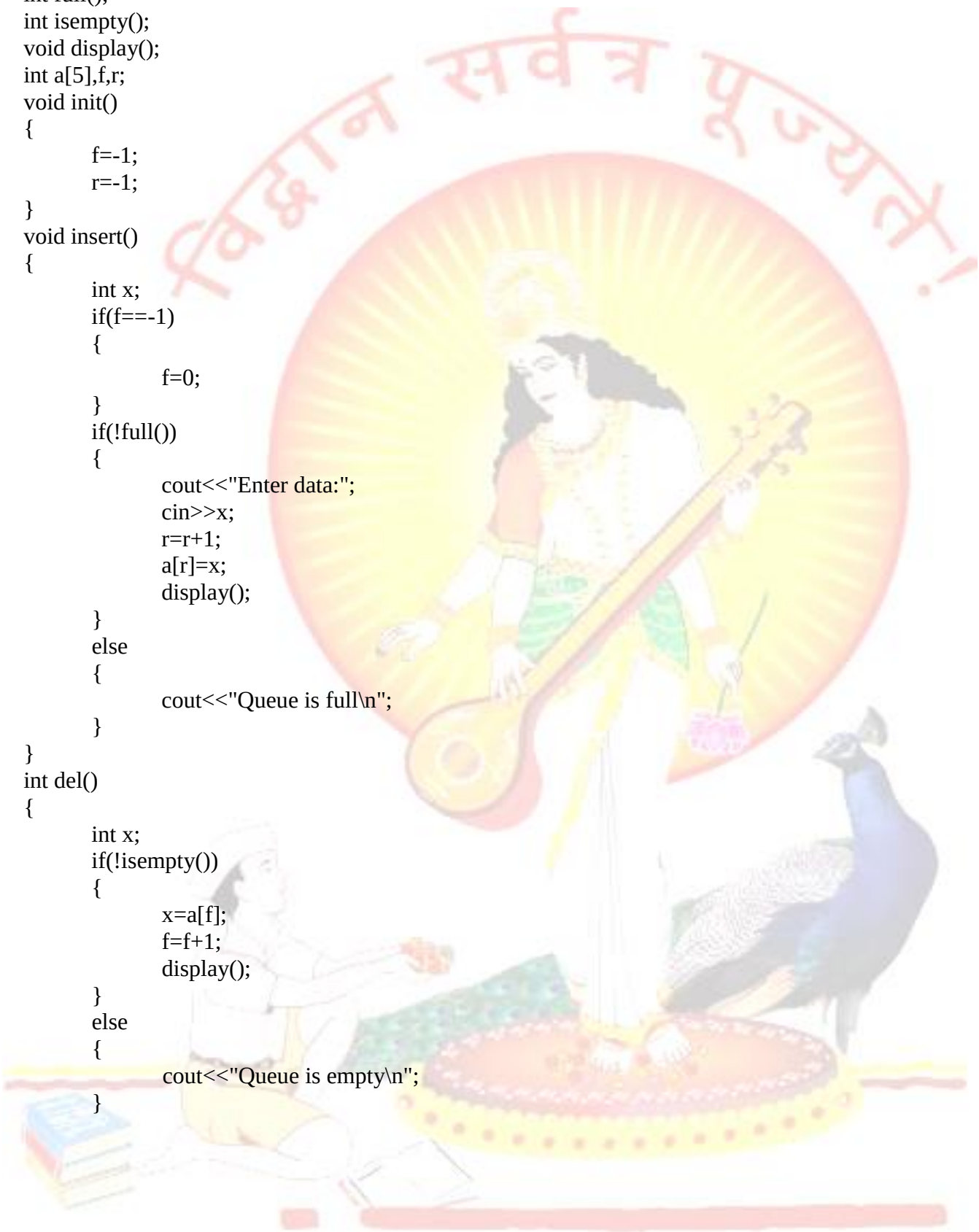
A	P	J	Total	Dated Sign
3	4	3	10	

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.

Program :

```
#include<iostream.h>
#include<conio.h>
void init();
void insert();
int full();
int isempty();
void display();
int a[5],f,r;
void init()
{
        f=-1;
        r=-1;
}
void insert()
{
        int x;
        if(f==-1)
        {
                f=0;
        }
        if(!full())
        {
                cout<<"Enter data:";
                cin>>x;
                r=r+1;
                a[r]=x;
                display();
        }
        else
        {
                cout<<"Queue is full\n";
        }
}
int del()
{
        int x;
        if(!isempty())
        {
                x=a[f];
                f=f+1;
                display();
        }
        else
        {
                cout<<"Queue is empty\n";
        }
```



```
}
int full()
{
        if(r==4)
                return 1;
        return 0;
}
int isempty()
{
        if(f==-1 || f==r+1)
                return 1;
        return 0;
}
void display()
{
        int i;
        cout<<"\nQueue=";
        for(i=f;i<=r;i++)
        {
                cout<<"\t"<<a[i];
        }
}
int main()
{
        clrscr();
        int ch,x;
        init();
        while(1)
        {
                cout<<"\n\nMenu\n1.Insert\n2.Delete\n3.Exit\nEnter choice:";
                cin>>ch;
                switch(ch)
                {
                        case 1:
                                insert();
                         break;
                        case 2:
                                del();
                         break;
                        case 3:
                                return 0;
                }
        }
}
```

Practical No:11(D)

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

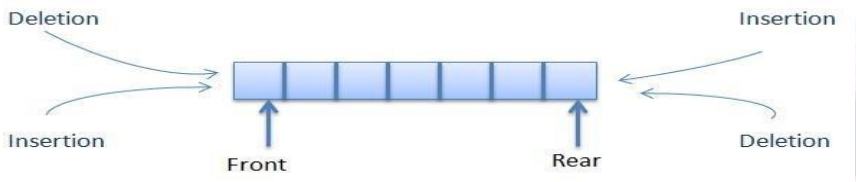
Outcome:

- 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

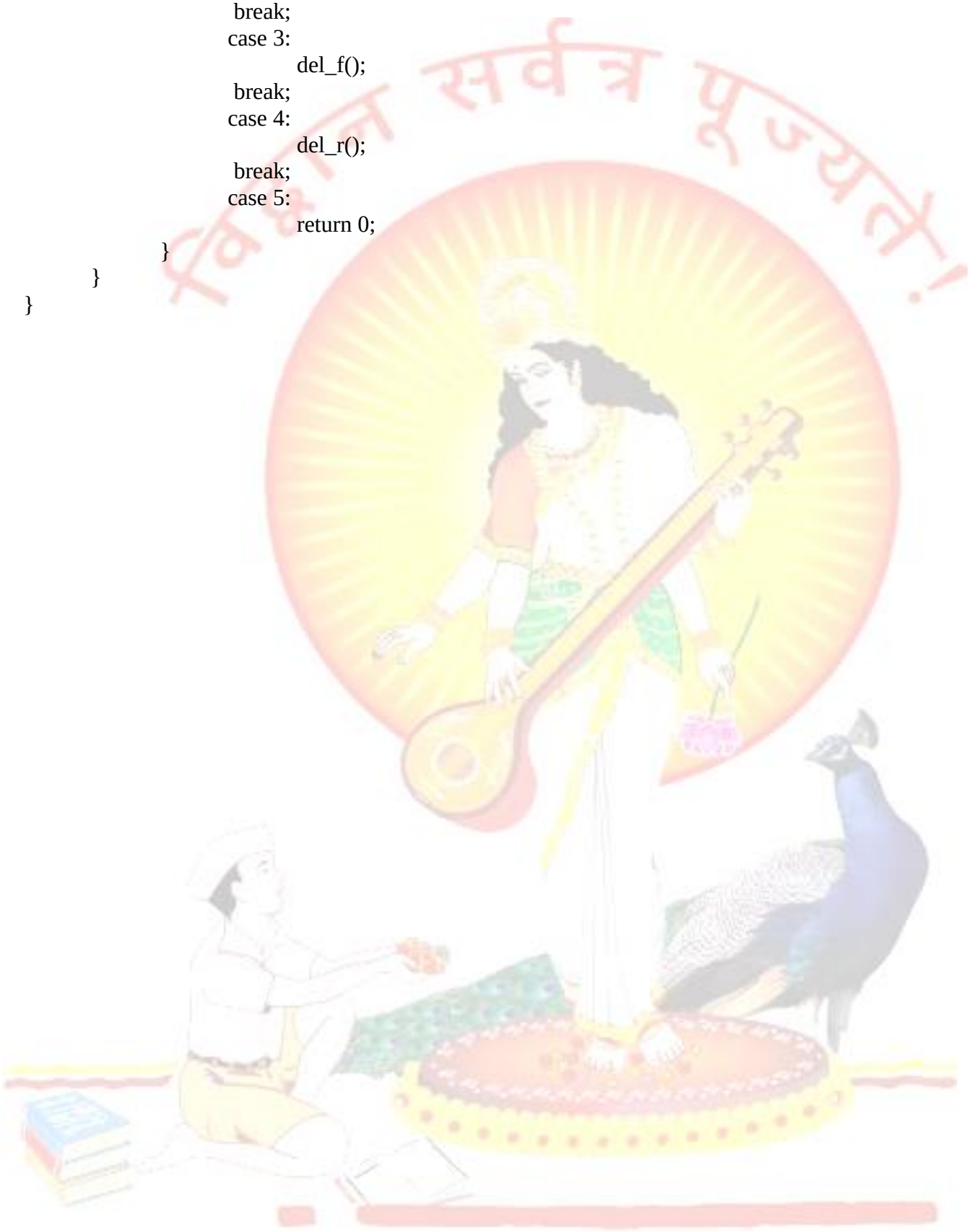
Program :

```
#include<iostream.h>
#include<conio.h>
void init();
void insert_r();
void insert_f();
int del_r();
int del_f();
int full_r();
int full_f();
int isempty();
void display();

int a[5],f,r,size=5;

void init()
{
        f=-1;
        r=-1;
}
void insert_r()
{
        int x;

        if(!full_r())
        {
                cout<<"Enter data:";
                cin>>x;
                r=r+1;
                a[r]=x;
        if(f==-1)
        {
                f=0;
        }
                display();
        }
        else
        {
                cout<<"Queue is full\n";
        }
}
void insert_f()
{
        int x;

        if(!full_f())
        {
                cout<<"Enter data:";
                cin>>x;
                if(f==-1)
                {
```

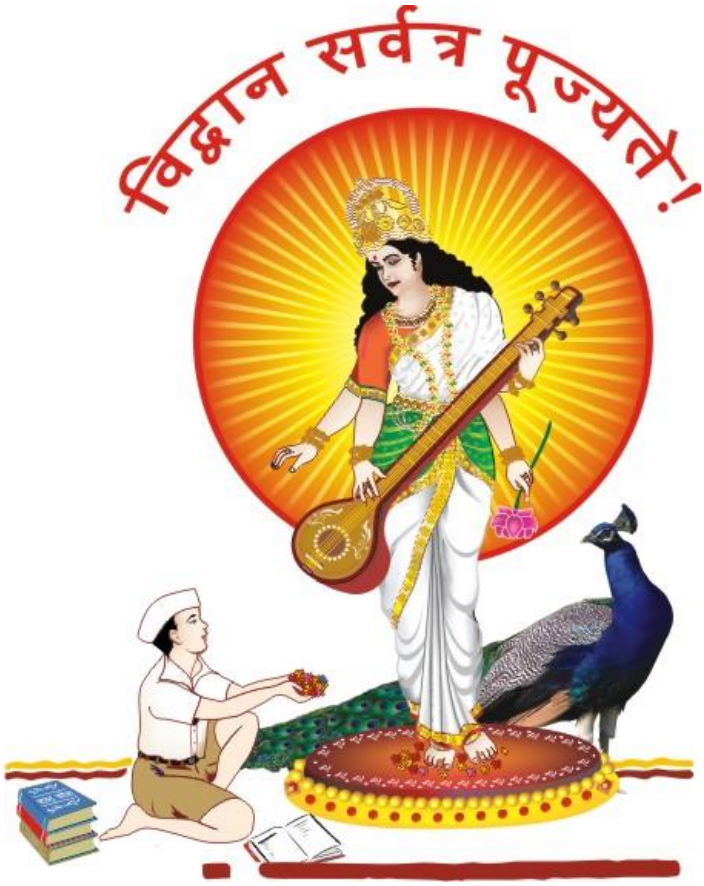
```
                f=r=0;
            }
            else
            {
              f=f-1;
            }
            a[f]=x;
            display();
        }
        else
        {
            cout<<"Queue is full\n";
        }

}
int del_f()
{
        int x;
        if(!isempty())
        {
            x=a[f];
            cout<<"\n Element deleted = "<<x;
            if(f==r)
            {
                f=-1;
                r=-1;
            }
            else
            {
              f=f+1;
              display();
            }
        }
        else
        {
            cout<<"Queue is empty\n";
        }
        return x;
}
int del_r()
{
        int x;
        if(!isempty())
        {
            x=a[r];
            cout<<"\n Element deleted = "<<x;
            if(r==f)
            {
              f=-1;
              r=-1;
            }
            else
```

```
                {
                  r=r-1;
                  display();
                }
        }
        else
        {
                cout<<"Queue is empty\n";
        }
        return x;
}
int full_f()
{
        if(f==0)
                return 1;
        return 0;
}
int full_r()
{
        if(r==size-1)
                return 1;
        return 0;
}
int isempty()
{
        if(f==-1)
                return 1;
        return 0;
}
void display()
{
        int i;
        cout<<"\nQueue=";
        for(i=f;i!=r;i=i+1)
        {
                cout<<"\t"<<a[i];
        }
        cout<<"\t"<<a[i];
}
int main()
{
        clrscr();
        int ch,x;
        init();
        while(1)
        {
                cout<<"\n\nMenu\n1.Insert at Front\n2.Insert at Rear"
                "\n3.Delete at Front\n4.Delete at rear"
                "\n5.Exit\nEnter choice:";
                cin>>ch;
                switch(ch)
                {
```

```
                        case 1:
                                insert_f();
                         break;
                        case 2:
                                insert_r();
                         break;
                        case 3:
                                del_f();
                         break;
                        case 4:
                                del_r();
                         break;
                        case 5:
                                return 0;
                }
        }
}
```



GROUP - E



GROUP - E

Practical No:12(E)

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

Aim: Write C++ program to store first year percentage of students in array. Write function for sorting array of floating point numbers in ascending order using a) Selection Sort b) Bubble sort and display top five scores.

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.

- Sorted list of elements
- 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:

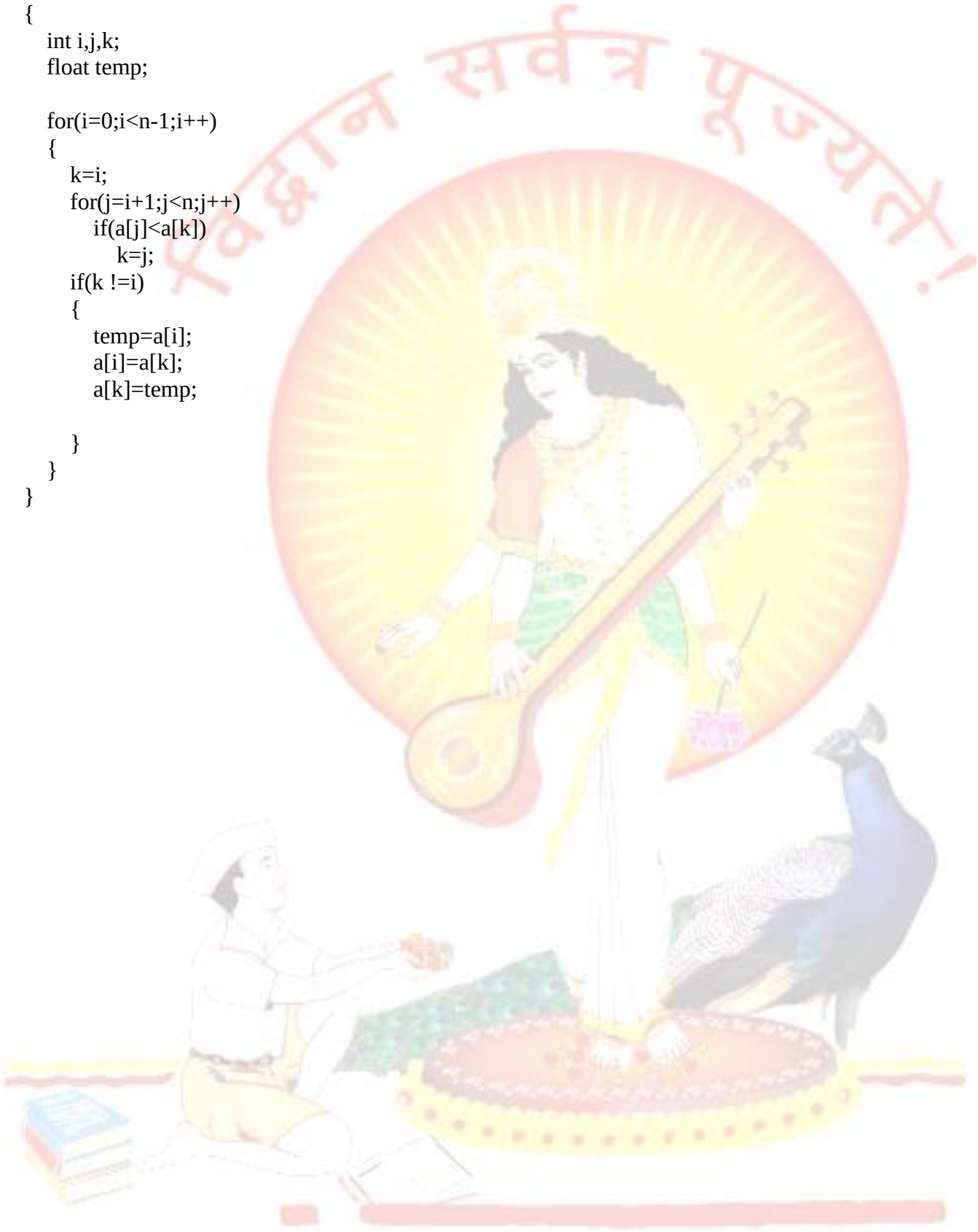
1. Explain the sorting?
2. What are the different types of sorts in data structures
3. Define the bubble sort?
4. Define the selection sort?
5. How many passes are required in selection sort?
6. What is the time complexity of selecion and bubble sort?

Program :

Program:

```
#include<stdio.h>

void selection(float[],int);
void bubble(float[],int);
int main()
{
   int n,i,op;
   float a[30];

   do
   {
      printf("\n 1)Bubble sort \n 2)Selection sort \n
      3)Quit"); printf("\n Enter your choice :"); scanf("%d",
      &op);
      if(op==1)
      {
         printf("\n Enter no. of elements :");
         scanf("%d",&n);
         printf("\n Enter array elements :");
         for(i=0;i<n;i++)
            scanf("%f",&a[i]);
         bubble(a,n);
         printf("\n Sorted array is :");
         for(i=0;i<n;i++)
            printf("%7.2f",a[i]);
      }
      if(op==2)
      {
         printf("\n Enter no. of elements :");
                  scanf("%d",&n);
                  printf("\n Enter array elements :");
                  for(i=0;i<n;i++)
                     scanf("%f",&a[i]);
                  selection(a,n);
                  printf("\n Sorted array is :");
                  for(i=0;i<n;i++)
                     printf("%7.2f",a[i]);
      }
   }while(op!=3);
}


void bubble(float a[], int n)
{
   int i,j;
   float temp;
   for(i=1; i<n;i++)
      for(j=0;j<n-i;j++)
         if(a[j]>a[j+1])
         {
            temp=a[j];
            a[j]=a[j+1];
            a[j+1]=temp;
```

```
        }
}


void selection(float a[],int n)
{
   int i,j,k;
   float temp;

   for(i=0;i<n-1;i++)
   {
      k=i;
      for(j=i+1;j<n;j++)
         if(a[j]<a[k])
            k=j;
      if(k !=i)
      {
         temp=a[i];
         a[i]=a[k];
         a[k]=temp;

      }
   }
}
```



Practical No:13(E)

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

Aim: Write C++ 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:

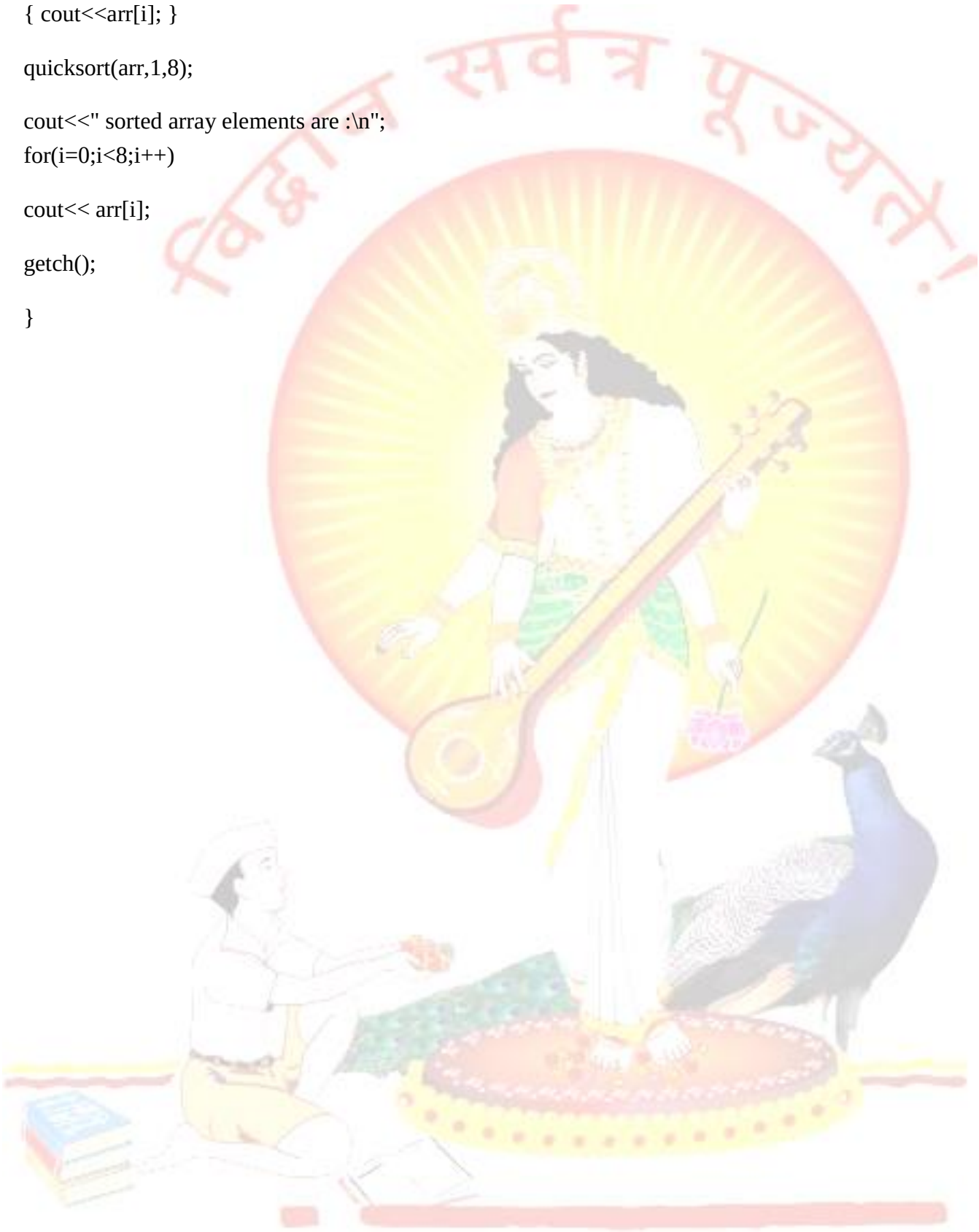
1.Explain the sorting?
2.What are the different types of sorts in data structures?
3.Define the quick sort?
5.How many passes are required in quick sort?
6.What is the time complexity of quick sort?

Program :

```
#include<iostream.h>
#include<conio.h>
```

```
#include<stdio.h>

int partition(int arr[], int p,int r);
void quicksort(int arr[],int p,int r)

{ int q;
if (p<r)
{ q= partition(arr,p,r);

quicksort(arr,p,q-1);

quicksort(arr,q+1,r);

}

int partition(int arr[],int p, int r)

{    int t, k,y,i,x;
x=arr[r];

i=p-1;

for(j=p;j<r;j++)

{ if(arr[j]<=x)

{ i=i+1;

t=arr[i];

arr[i]=arr[j];

arr[j]= t;

}

}

k=arr[i+1];

arr[i+1]=arr[r];

arr[r]=k; y

= i+1;

return y ; }

void main()

{

int arr[8],i;

cout<<"Enter first year percentage of students : \n";

for(i=0;i<8 ;i++)
```

```
{ cin>>arr[i]; }

cout<<"\n first year percentage of student is : \n";

for(i=0;i<8;i++)

{ cout<<arr[i]; }

quicksort(arr,1,8);

cout<<" sorted array elements are :\n";
for(i=0;i<8;i++)

cout<< arr[i];

getch();

}
```



Practical No :14 (E)

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

Aim: Write C++ 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:

- Knowledge of sorting techniques

Objective:

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

- Sorted list of elements
- 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

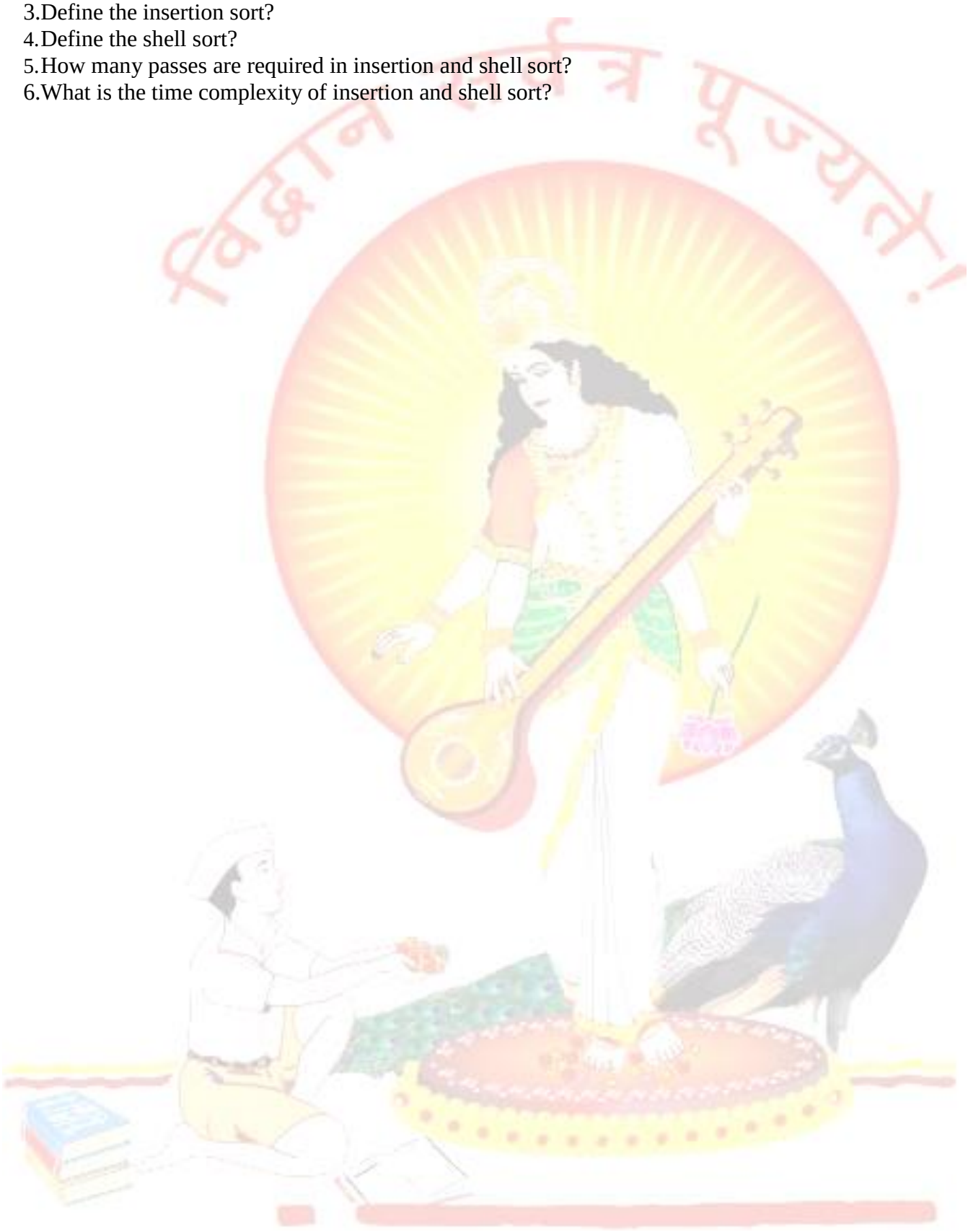
Conclusion:

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:

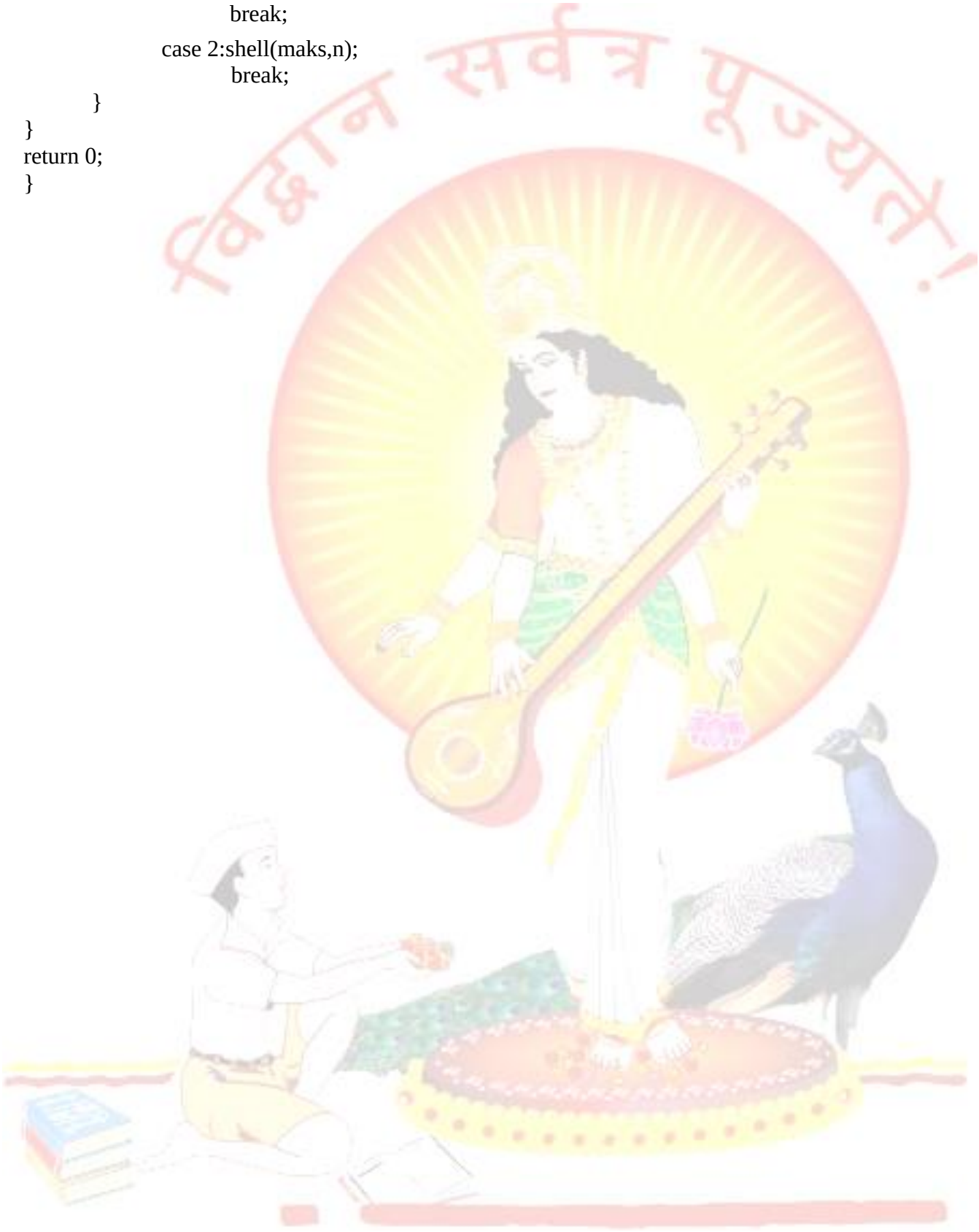
1. Explain the sorting?
2. What are the different types of sorts in data structures
3. Define the insertion sort?
4. Define the shell sort?
5. How many passes are required in insertion and shell sort?
6. What is the time complexity of insertion and shell sort?



Program :

```
#include<iostream>
using namespace std;
void in(float maks[20],in n))
{
        in i,j;
        float temp;
        for(i=1;i<n;i++)
        {
                temp=maks[i];
                for(j=i-1;j>=0&& maks[j]>temp;j--)
                marks[j+1]=marks[j];
                marks[j+1]=temp;
        }
        cout<<"\n Tiop five scores are:"
        for(i=n-1;i>=n;i--)
        cout<<"\t"<<marks[i];
}

void shell(float marks[20],int n)
{
        int i,j,step;
        float temp;
        for(step=n/2;step>0;step=step/2)
        for(i=step ;i<n;i++)
        {
                temp=marks[i];
                for(j=i;j>=step;j=j-step)
                if(temp<marks[j-step])
                marks[j]=marks[j-step];
                else
                break;
                marks[j]=temp;
        }
        cout<<"\n Top five scores are:";
        for(i=n-1;i>=n-5;i--)
        cout<<"\t"<<marks[i];
}

int main()
{
        float marks[20];
        int i,n,ch;
        cout<<"\n Enter the total no.of students:";
        cin>>n;
        for(i=0;i<n;i++)
        {
                cout<<"\n Ener the percentage marks o second year student"<<i+1<<"::"; cin>>marks[i];


        }

while(ch!=3)
{
```

```
cout<<"\n1. Insertion sort 2. Shell sort 3.Exit";
cout<<"\n Ener your choice:";
        cin>>ch;
        switch(ch)
        {
                case 1:in(maks,n);
                        break;
                case 2:shell(maks,n);
                        break;
        }
  }
  return 0;
  }
```



THANKS..!

PROF. ANAND NANDLAL GHARU

ASSISTANT PROFESSOR

PVGCOE, NASHIK

Blog : anandgharu.wordpress.com