ALAGAPPA UNIVERSITY

**[Accredited with 'A+' Grade by NAAC (CGPA:3.64) in the Third Cycle
and Graded as Category–I University by MHRD-UGC]**

KARAIKUDI – 630 003

DIRECTORATE OF DISTANCE EDUCATION

Vallal Dr.RM.Alagappa Chettiar

Master of Computer Applications
315 14

LAB: DATA STRUCTURE USING C++

I - Semester

ALAGAPPA UNIVERSITY

Directorate of Distance Education

Master of Computer Applications

I - Semester

315 14

LAB: DATA STRUCTURE USING C++

Author

Dr. Kavita Saini, *Assistant Professor, School of Computer Science & Engineering, Galgotias University, Greater Noida*

SYLLABI-BOOK MAPPING TABLE
Lab: Data Structure Using C++

Syllabi	Mapping in Book

BLOCK 1: SIMPLE C++ PROGRAMS

1. **Introduction:** Simple C++ Programs
2. **Control Structures:** Using if and switch constructs Programs
3. **Looping, Arrays, Structure statements:** for, while, do-while, Strings and Matrices Programs Problems

BLOCK 2: OOPs CONCEPTS

4. **Functions:** static function, friend function, constructor, destructor and operator overloading and Recursive programs
5. **Inheritance and polymorphism:** Inheritance types and polymorphism types, Virtual function
6. **File:** File Handling C++ Programs, opening and closing a data file - creating a data file, processing a data file
7. **Pointers:** Pointers and Pointers with Arrays Programs

BLOCK 3: LINEAR DATA STRUCTURE

8. **Stacks :** Stack Implementation, expression evaluation, Polish notation
9. **Queues:** Queue Implementation, Applications of Queue
10. **Linked List programs:** List, Merging lists, Linked list, Single linked list, Double Linked List, Header Linked list, Insertion and Deletion of linked list, Traversing a linked list

BLOCK 4: NON LINEAR DATA STRUCTURE

11. **Tree Programs :** Trees, Binary Trees, Types of Binary trees, Binary Tree Representation, Traversing Binary Trees, Binary Search tree, Insertion and Deletion operations
12. **Graphs:** Shortest Path Algorithms
    o Dijkstra's Algorithm
    o Graphs with Negative Edge costs
    o Acyclic Graphs
    o All Pairs Shortest Paths Algorithm Minimum cost Spanning Trees
    o Kruskal's Algorithm
    o Prims's Algorithm
    o Applications

BLOCK 5: SEARCHING AND SORTING ALGORITHMS

13. **Searching Techniques:** Linear and Binary search Programs
14. **Sorting techniques:** Bubble sort, Quick sort, Insertion sort, Merge sort

INTRODUCTION

Object-oriented Programming (OOP) is one of the most impressive programming paradigms in software development. C++ has become one of the most popular OOP languages used for developing real-world applications. C++ is a programming language that extended from the ubiquitous C language. It treats data as a crucial element—not allowing it to move freely around the system. Therefore, the main emphasis in C is on data and not on the procedure. You can design programs around the data being operated upon in C++. An object-oriented language helps in combining data and functions that operate on data into a single unit known as object. C++ is used for developing different types of applications, such as real-time systems, simulation modelling, expert systems. It also provides flexibility to a user to introduce new types of objects in his programming on the basis of the requirement of the application.

This lab manual, *Lab: Data Structure using C++*, contains several programs based on C++ concepts, such as classes, inheritance, stack and queue, to provide the concept of programming. In addition, it will help students in coding and debugging their programs. The manual provides all logical, mathematical and conceptual programs that can help to write programs very easily in C++ language. These exercises shall be taken as the base reference during lab activities for students of MCA. There are also many Try Yourself Questions provided to students for implementation in the lab.

BLOCK I   SIMPLE C++ PROGRAMS

Introduction

C++ language is invented by Bjarne Stroustrup in 1980 at Bell Laboratories, New Jersey. C++ language was initially called "C with Classes" but in 1983 this name was changed to C++. C++ is a superset of C.

The purpose of C++ is to overcome this limit and provide a better way to manage larger, more complex programs, by using object oriented programming (OOP). C++ is very popular language as it has many features as mentioned below:

- Classes and objects
- Encapsulation
- Information hiding
- Inheritance
- Polymorphism

Portable language

C++ is a portable language having feature of carrying the instruction from one system to another system. In C++ Language **.cpp** file contain source code, we can edit also this code. **.exe** file contain application, only we can execute this file. When we write and compile any C++ program on window operating system that program easily run on other window based system.
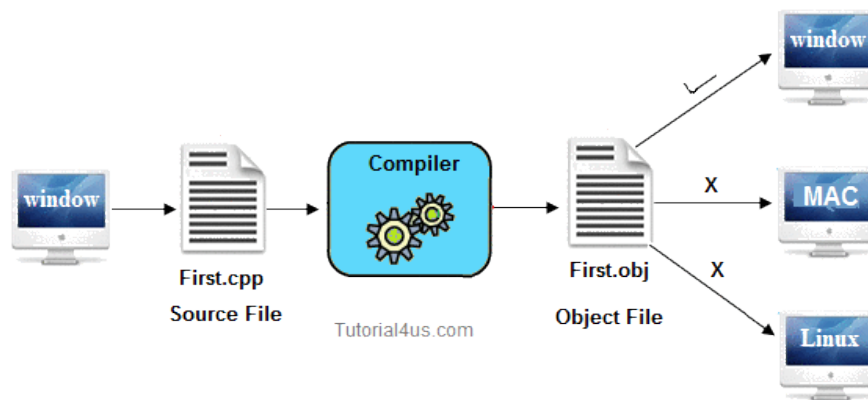
Fig. 1.1 *Representing the C++ object file running on Windows.*

Recommended System/Software Requirements:

1. Intel based desktop PC of 166MHz or faster processor with at least 64 MB RAM and 100 MB free disk space.

2. Turbo C++ compiler or GCC compilers

In this manual we have used Turbo C++. To write C++ code first we need to open Torbo C++.

For every C++ program, we need to follow the steps given below for writing and executing a program:

Write a program code-> save your program (F2)->compile (Alt+F9)-> Run(Ctrl +F9)

Step 1: Click on Turbo C++ from start menu or double click on Turbo C++ on desktop.

After clicking on Turbo C++ following screen will appear:
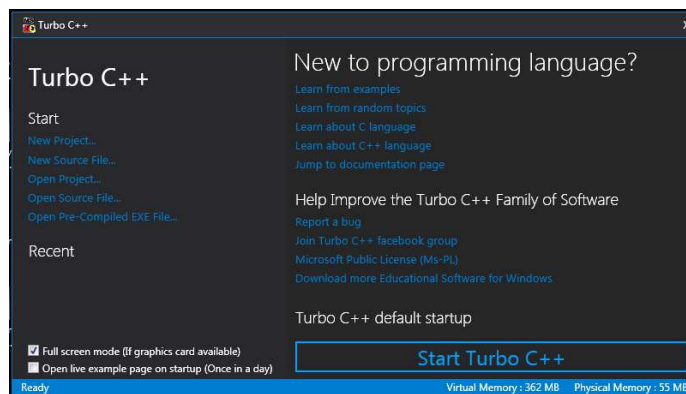
Step 2: Click on Start Turbo C++. After clicking on Start Turbo C++ button following screen will appear:
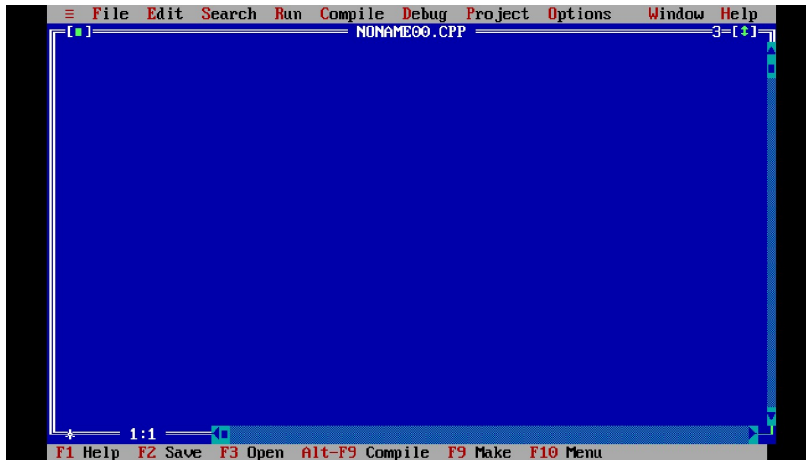
This is the editor where we will write code of C++ programs.

Step 3: Write a program to print "Hello" on screen (Hello.cpp).
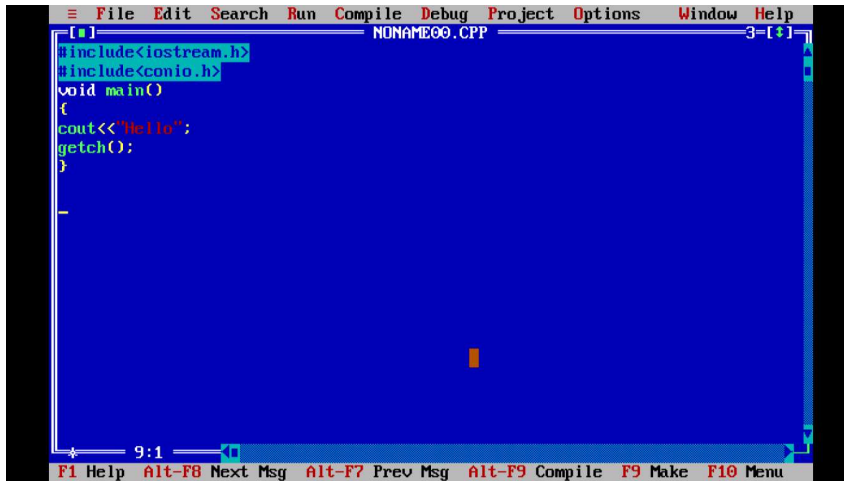
Step 4: Save program by name hello.cpp by pressing F2 key or by using menu option File->Save As:
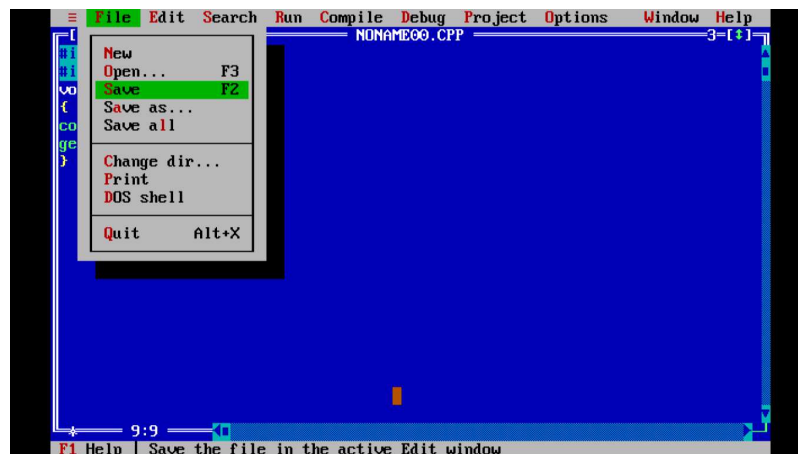
Step 5: Compile program by hello.cpp by pressing Alt+F9 keys or by using menu option Compile-> Compile**:**
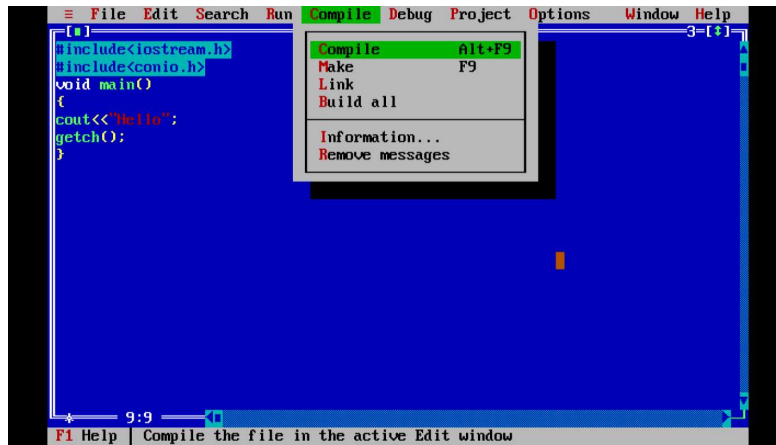
Step 6: Run program by hello.cpp by pressing Ctrl +F9 keys or by using menu option Run->Run.
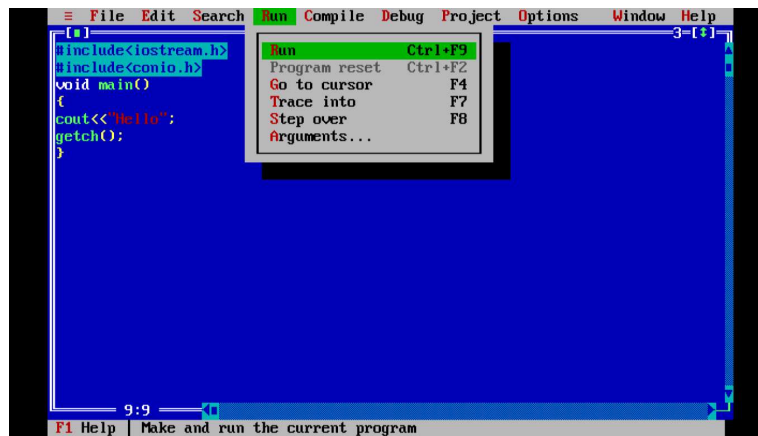
Output:
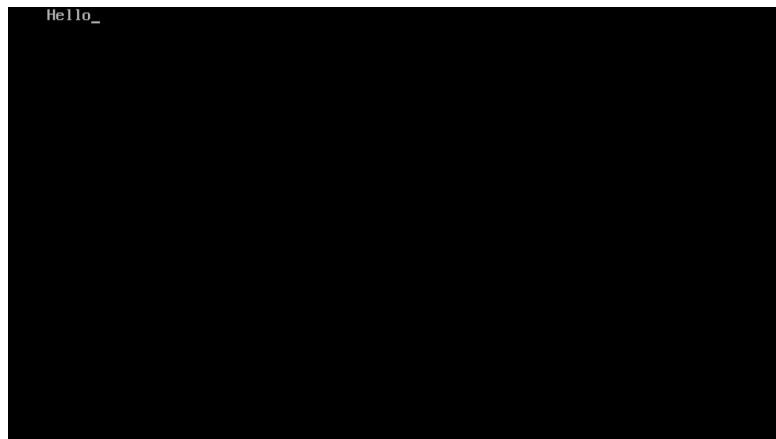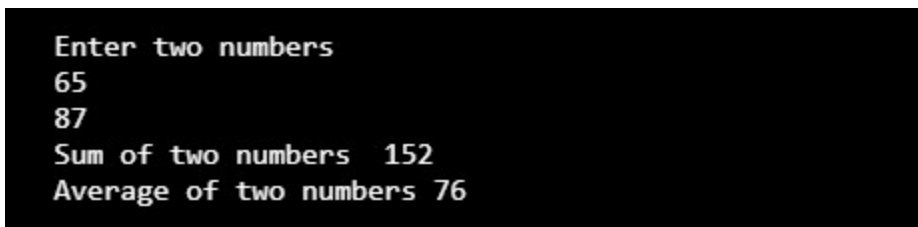
Simple C++ Programs

1. Write a program to take two numbers as input and print their sum and average.

```
//Program to take input in two numbers and print sum and
average
#include<iostream.h>
 void main()
{
int  num1,num2,sum,avg;

cout<<"Enter two numbers"<<endl; //output statement
cin>>num1; //input statement
cin>>num2;
sum=num1+num2;
avg=sum/2;
cout<<"Sum of two numbers  "<<sum<<endl;
cout<<"Average of two numbers "<<avg;
}
```

Output:

```
Enter two numbers
65
87
Sum of two numbers   152
Average of two numbers 76
```

2. Write a program to swap two numbers without using a third Variable.

```
#include <iostream.h>
void main()
{
int  num1,num2;
cout<<"Enter two numbers"<<endl;
cin>>num1>>num2;
num2 = num1+num2;
num1 = num2 - num1;
num2 = num2 - num1;
cout<<"values after swaping :\n";
cout<<"Value of a Num1 "<<num1<<endl;
cout<<"Value of a Num2 "<<num2<<endl;
}
```

Output:

```
Enter two numbers
3
4
values after swaping :
Value of a Num1 4
Value of a Num2 3
```

Try yourself:

  (i) Write a program to calculate volume of cylinder.

```
Volume of cylinder= PI*r*r*h
```

 (ii) Write a program to calculate curved surface area of cylinder.

```
Curved surface area of cylinder= 2*PI*r*h
```

(iii) Write a program to print ASCII value of digits, uppercase and lowercase
      alphabets.

Control Structures: Using if and switch constructs Programs:

3. Write a program to check whether the number provided is even or odd.

```
#include <iostream.h>
void main()
{
int  num;
cout<<"Enter a number: ";
cin>>num;
if(num%2==0)
{
cout<<"Number is even ";
}
else
{
cout<<"Number is odd ";
}


}
```

Output:

```
Enter a number: 2
Number is even
```
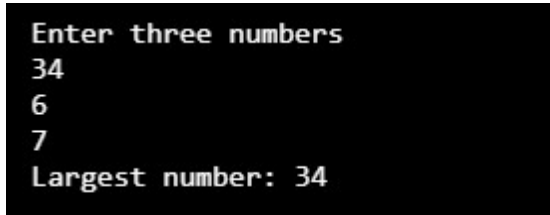
**4. Write a program to print the largest number among three numbers given
by the user.**

```
// program to print the largest number among three numbers
#include <iostream.h>
 void main()
{
int num1,num2,num3;
cout<<"Enter three numbers"<<endl;
cin>>num1>>num2>>num3;

if(num1 >= num2 && num1 >= num3)
    {
        cout << "Largest number: " << num1;
    }
   else  if(num2 >= num1 && num2 >= num3)
     {
        cout << "Largest number: " << num2;
     }
    else
    {
        cout << "Largest number: " << num3;
    }

}
```

Output:

```
Enter three numbers
34
6
7
Largest number: 34
```

5. Write a single program that provides the sum, difference, multiplication and division of two numbers.

```
#include <iostream.h>
void main()
{    int num1, num2; char op;
        cout << "Enter two numbers: ";
    cin >> num1 >> num2;
    cout << "Enter operator : ";
    cin >> op;
      switch (op)
      {
```

```
       case '+': cout <<"\n Sum of two numbers is: "<<
num1+num2;
     break;
       case '-': cout <<"\n Subtraction of two numbers
is: "<< num1- num2;
       break;
     case '*': cout <<"\n Multiplication of two numbers
is: "<< num1*num2;
        break;
       case '/': cout <<"\n Division of two numbers is:
"<< num1/num2;
       break;
       default: cout << "\n Invalid operator";
    break;
     }
}
```

Output:

```
Enter two numbers: 45
10
Enter operator: *
Multiplication of two numbers is: 450
```

> **Try yourself:**
>
> (1) Write a program to convert a lowercase alphabet to uppercase or vice-versa.
>
> (2) Write a program to check whether a year is leap year or not.
>
> (3) Write a program to check whether a given character is uppercase or lowercase alphabet or a digit or a special character.

L**ooping, Arrays, Structure statements: for, while, do-while, Strings and Matrices Programs.**

6. Write a program to print table of any number using for loop.

```
// program to print table of any number

#include <iostream.h>
void main()
{
   int num, i;
   cout<<"Enter a number: ";
   cin>>num;
 cout<<"Table of "<<num<<endl;
for (i=1;i<=10;i++)
```

```
{
cout<<num*i<<endl;
}
}
```

Output:

```
Enter a number: 9
Table of 9
9
18
27
36
45
54
63
72
81
90
```

7. Write a program to print Fibonacci Series (0, 1, 1, 2, 3, 5, 8, 13, 21,...)

```
// Program to print Fibonacci Series using for loop
#include <iostream.h>
void main()
{
    int num, i, a=0, b=1, c;
     cout<<"Enter a number of terms for Series: ";
    cin>>num;
     cout<<"Fibonacci series : \n";
for (i=0; i<num; i++)
{
cout<<"\n"<<a;
c=a+b;
a=b;
b=c;
}
}
```

Output:

```
Enter a number of terms for Series: 9
Fibonacci series :

0
1
1
2
3
5
8
13
21
```

8. Write a program to check the number is Armstrong or not.

A number is known as **Armstrong number** if sum of the cubes of its digits is equal to the number itself.

For example

370 is an Armstrong number because:

$$370 = 3*3*3 + 7*7*7 + 0*0*0$$

$$= 27 + 343 + 0$$

$$= 370$$

```cpp
// C++ Program to check Armstrong Number
#include <iostream.h>

void main()
{
    int num, sum = 0, rem,temp;
    cout<<"Enter a number: ";
    cin>>num;

     temp =num;
while (num>0)
{
rem= num%10;
sum= sum+(rem*rem*rem);
num= num/10;
}
if (temp==sum)
cout<<"Number in armstrong "<<endl;
else
cout<<"Number is not armstrong ."<<endl;
}
```

Output:

```
Enter a number: 370
Number in armstrong
```

9. Write a C++ program to print table of a number using do while loop.

```cpp
//C++ program to print table of any number using do while
loop
#include <iostream.h>
void main()
```

```
{
   int num, i;
       cout << "Enter any number: ";
   cin >> num;
   cout<<"\n Table of" <<num<<endl;
   i=1;
   do{
     cout<<num*i<<endl;
      i++;
   }while (i<=10);
}
```

Output:

Try yourself:

(1) Write a program to reverse a number.

(2) Write a program to check whether a number is prime number or not.

(3) Write a program to convert binary number to decimal number.

10 Write a program that takes values in an array and also display them.

```
//C++ program to scan and print values using array
#include <iostream.h>
int main()
{
   int arr[5],i;
       cout << "Enter 5 numbers:\n ";
           for (i=0;i<5;i++)
       cin >> arr[i];
   cout<<"\n Array values are "<<endl;
     for (i=0;i<5;i++)
       cout<<arr[i]<<endl;
}
```

Output:

11. Write a program that take a string as input and print it.

```cpp
//C++ Program to take input in string and print
#include <iostream.h>
#include <conio.h>
void main()
{
char str[15];
cout<<"Enter your name: ";
    cin>>str;
    cout<<"\n Welcome "<<str;
getch ();
}
```

12. Write a program to print length of a string provided.

```cpp
//C++program to count string length
#include<iostream.h>
void main( )
{
    int i,count=0;
    char str[50];
    cout<<"Enter any string  ";
     cin.getline (str, 50);  //getline function allows
user to input string with space
    //loop will run till it reaches to string terminator
'\0'
    for (i = 0; str[i] != '\0'; i++)
    {
    count ++;
    }
    cout << "\n Length of string is  " << count;
}
```

Output:

```
Enter any string  programming

Length of string is  11
```

13. **Write a program to check whether a string is palindrome or not.**

```cpp
// C++ program to check a string is palindrome or not
#include<iostream>
using namespace std;
int main( )
{
    int i,len=0;
    char str[50],rev_str[50];

    cout<<"Enter any string  ";
    cin.getline(str, 50);  //getline function allows user
to input string with space
 //count length of string
    for (i = 0; str[i] != '\0'; i++)
    {
    len++;
    }
    //copy str to rev_str
    int j=0;
    for (i = len - 1; i >= 0 ; i—,j++)
    {
        rev_str[j] = str[i];
    }
    rev_str[j] ='\0' ;  //reverse string is terminated
    //compare both strings
    int flag=0;
    for (i = 0; i < len ; i++)
    {
        if (str[i]==rev_str[i])
             flag = 1;
            else
            {
            break;      //exit from loop
            }
    }
```

```
          if (flag == 1)
     cout<<" \n string is a palindrome";
              else
     cout<<" \n string is a not palindrome";
  }
```

Output:

```
Enter any string  nitin

string is a palindrome
```

14. Write a program to print the largest number in an array.

```cpp
//C++ program to print the largest number in an array
#include <iostream.h>
int main()
{
    int arr[5], i, max;
        cout << "Enter 5 numbers:\n ";
     for (i=0;i<5;i++)
        cin >> arr[i];
 max= arr[0];
     for (i = 1;i < 5; i++)
     {
           if (max < arr[i])
          max = arr[i];
     }
    cout << "Largest element = " << max;
}
```

Output:

```
Enter 5 numbers:
 65
 4
 5
 76
 4
 Largest element = 76
```

Try yourself

(1) Write a program to insert an element in an array.

(2) Write a program to find sum of elements of an array.

(3) Write a program to find largest number from an array.

15. Write a program that provides the sum of two matrices.

```cpp
//C++ program to print sum of two matrices
#include<iostream.h>
int main()
{
   int i, j, m1[10][10], m2[10][10], sum[10][10];
   cout << "Enter the elements of first matrix\n";
    for (  i = 0; i < 3; i++ )
    {
       cout<<"\n enter values for row  "<<i+1<<endl;
       for ( j = 0 ; j<3 ; j++ )
    {    cin >> m1[i][j];}
    }

   cout << "Enter the elements of second matrix\n";

   for (  i = 0 ;i < 3; i++ )
    {
        cout<<"\n enter values for row  "<<i+1<<endl;
      for ( j = 0 ; j< 3 ; j++ )
    {   cin >> m2[i][j];
     }
    }
   cout << "Sum of two matrices \n";
   for (  i = 0 ;i < 3 ; i++ )
   {
      for ( j = 0 ; j<3 ; j++ )
   {     sum[i][j] = m1[i][j]+m2[i][j];
    cout << sum[i][j] << "\t";

    }
      cout<<endl;
   }
   }
```

Output:

16. Write a program to find out the product of two matrices.

```cpp
//C++ program for matrix multiplication
#include<iostream.h>
int main()
{
   int i,j,k, m1[10][10], m2[10][10], res[10][10];
   cout << "Enter the elements of first matrix\n";
    for (  i = 0 ;i < 3 ; i++ )
    {
       cout<<"\n enter values for row  "<<i+1<<endl;
      for ( j = 0 ; j<3 ; j++ )
    {     cin >> m1[i][j];}
    }
   cout << "Enter the elements of second matrix\n";
   for (  i = 0 ;i < 3; i++ )
    {
        cout<<"\n enter values for row  "<<i+1<<endl;
       for ( j = 0 ; j< 3 ; j++ )
      {   cin >> m2[i][j];
       }
     }
    for (i = 0; i < 3; ++i)
       {
```

```
        for (j = 0; j < 3; ++j)
    {
         res [i][j]=0;
        for (k = 0; k < 3; ++k)
        {
            res[i][j] += m1[i][k] * m2[k][j];
        }
    }
  }
cout << "Multiplication of two matrices \n";
   for ( i = 0 ;i < 3 ; i++ )
   {
      for ( j = 0 ; j<3 ; j++ )
     {
   cout << res[i][j] << "\t";
    }
      cout<<endl;
   }
   }
```

Output:

```
Enter the elements of first matrix

 enter values for row  1
1 2 3

 enter values for row  2
1 2 3

 enter values for row  3
1 2 3
Enter the elements of second matrix

 enter values for row  1
2 3 4

 enter values for row  2
2 3 4

 enter values for row  3
2 3 4
Multiplication of two matrices
12  18  24
12  18  24
12  18  24
```

Try yourself:

(1) Write a program to print sum of diagonal values of a square matrix.

(2) Write a program to find highest and lowest element of a matrix.

(3) Write a program to convert first letter of each word of a string to uppercase and other to lowercase.

(4) Write a program to find substring in string (pattern matching).
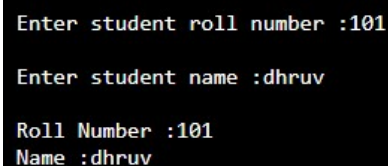
BLOCK II OOPs CONCEPTS

1. Write a program to demonstrate the use of class and object.

```cpp
//C++ sample program for class and object
#include<iostream.h>
 //class
class student
 {
    private:        //scope of variables is private
    //member variables
     int rno;
    char name[10];
    public:         //scope of functions is public
// member functions
void input()
{
    cout<<"\n Enter student roll number :";
    cin>>rno;
    cout<<"\n Enter student name :";
    cin>>name;
    }
void display()
{
    cout<<"\n Roll Number :"<<rno;
    cout<<"\n Name :"<<name;
 }
 } ;   //class closed
int main()
{
    student obj;         //object of student class
    obj.input();         //call of input function
    obj.display();       //call of display function
    }
```

Output:

```
Enter student roll number :101

Enter student name :dhruv

Roll Number :101
Name :dhruv
```

Keywords: private and public

You may have noticed two keywords: private and public in the above program.

The **private** keyword makes data and functions private. Private data and functions can be accessed only from inside the same class.

The **public** keyword makes data and functions public. Public data and functions can be accessed out of the class.

2. Write a program to demonstrate the use of constructor in a class.

```cpp
//C++ sample program for constructor
#include<iostream.h>
//class
 class student
 {
    private:        //scope of variables is private
    //member variables
     int rno;
    char name[10];
    public:         //scope of functions is public
    student()
     {
        cout<<"Constructor \n";
         rno=0;
     }
// member functions
void input()
{
    cout<<"\n Enter student roll number :";
    cin>>rno;
    cout<<"\n Enter student name :";
    cin>>name;
     }
void display()
{
    cout<<"\n Roll Number :"<<rno;
    cout<<"\n Name :"<<name;
 }
 } ;
int main()
{
```
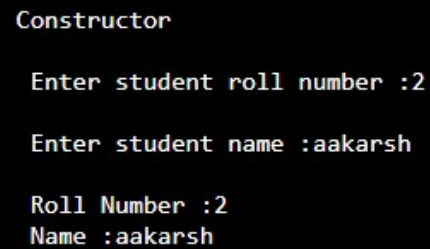
```
student obj;
obj.input();
obj.display();
}
```

Output:


```
Constructor

 Enter student roll number :2

 Enter student name :aakarsh

 Roll Number :2
 Name :aakarsh
```

3. Write a program to demonstrate the use of constructor and destructor in a class.

```cpp
//C++ sample program for constructor and destructor
#include<iostream.h>

//class

 class student
 {
    private:
    //member variables
     int rno;
    char name[10];

    public

     // constructor
    student()
     {
        cout<<"Constructor \n";
         rno=0;
     }

// member functions
void input()
{
    cout<<"\n Enter student roll number :";
    cin>>rno;
```

```
    cout<<"\n Enter student name :";
    cin>>name;
     }
```

```
void display()
{
    cout<<"\n Roll Number :"<<rno;
    cout<<"\n Name :"<<name;
 }

 //destructor
 ~student()
    {
        cout<<"\n Destructor \n";
            }
 } ;

int main()
{
   student obj;
   obj.input();
   obj.display();
    }
```

Output:


```
Constructor

 Enter student roll number :101

 Enter student name :Dhruv

 Roll Number :101
 Name :Dhruv
 Destructor
```

4. Write a program to demonstrate the use of static variable.

```
//C++ sample program for static variable
#include <iostream.h>
void test()
{
    // static variable
    static int count = 0;
    cout << count <<endl;
    count++;
```
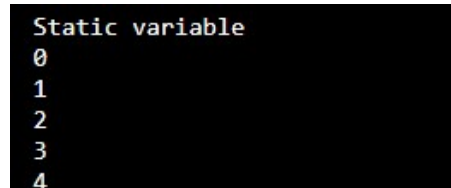
```
                   }
                   int main()
                   {
                        cout << "Static variable " <<endl;
                     for (int i=0; i<5; i++)
                         test();
                     }
```

Output:

```
Static variable
0
1
2
3
4
```

Features of static function

- Static member functions have a class scope and they do not have access to the 'this' pointer of the class.

- The main usage of static function is when the programmer wants to have a function which is accessible even when the class is not instantiated.

- A normal member function is accessed using the object and an operator called the dot member access operator.

- The functions declared static or static functions are accessed using only the class name and the scope resolution operator, unlike in normal member functions where these are not used.

5. Write a program to demonstrate the use of static variable and static function.

```
//C++ sample program for static variable and static function
#include <iostream.h>
class test
{
private:
    static int count;   //Static data
    int n;
public:
 //Constructor
    test ()
    {
    count = count+1;
        n=count;
    }
```

```
      //static function
   static void function1()
   {
      cout << "\nResult is: " << count<<endl;
   }
   //Normal  function
   void counter ()
   {
      cout << "\nCounter is: " << n<<endl;
   }
    //Destructor
   ~test ()
   {
      count = count-1;
   }
};
   int test::count=0;
 int   main()
{
   test obj1;
    //Static function is accessed using class name and
scope resolution operator (::)
   test::function1 ();
   test obj2,obj3,obj4;
   test::function1 ();
    //normal function is accessed using object name and
the dot member access operator(.)
   obj1.counter();
   obj2.counter();
   obj3.counter();
   obj4.counter();
}
```

Output:

```
Result is: 1

Result is: 4

Counter is: 1

Counter is: 2

Counter is: 3

Counter is: 4
```

6. Write a program to demonstrate the use of static function and variable.

```cpp
//C++ Program to count the object value using the keyword
static variable.
#include<iostream.h>
class static_class {
    int n;
    static int count; //static variable
public:
//constructor
    static_class()
    {
        n = ++count;
    }
    void obj_number()
{
        cout << "\n\tObject number is :" << n;
    }
    static void obj_count()
{
        cout << "\nNumber of Objects :" << count;
    }
};
int static_class::count;
int main()
{
    static_class obj1, obj2;

    obj1.obj_count();
    obj1.obj_number();
    obj2.obj_count();
    obj2.obj_number();
     return 0;
}
```

Try yourself:

(1) Write a program to swap two numbers using class

(2) Write a Program to Print Numbers From 1 to n using class

(3) Write a program to calculate area of a circle,a rectangle or a triangle depending upon user's choice using class

7. Write a program to get and print student data using inheritance.

```
// program to get and print student data using inheritance
#include <iostream.h>
//class
class student
 {
    private:        //scope of variables is private
    //member variables
     int rno;
    char name[10];
    public:         //scope of functions is public
// member functions
void input ()
{
    cout<<"\n Enter student roll number :";
    cin>>rno;
    cout<<"\n Enter student name :";
    cin>>name;
     }

void display ()
{
    cout<<"\n Roll Number :"<<rno;
    cout<<"\n Name :"<<name;
 }
 } ;   //class closed

 class fee:public student     //class fee(derived) class
is inheriting student (base) class
 {
     float fee;        //default scope in private

     public:
     void input_data ()
     {
     input ();      //call of input function of student
class
    cout<<"\n Enter Fee :";
    cin>>fee;
     }
```

```
    void display_data ()
      {
     display ();     //call of display function of student
class
      cout<<"\n Fee  :"<<fee;
      }
  };
int main()
{
   fee obj;          //object of fee class
   obj.input_data ();
   obj.display_data ();
    }
```

Output:

```
Enter student roll number :101

Enter student name :dhruv

Enter Fee :20000

Roll Number :101
Name :dhruv
Fee  :20000
```

8. Write a program to overload a sum function.

```
#include <iostream.h>
class Test
{
    public:
 int sum(int a,int b)
{
       return a + b;
    }
 int sum (int a, int b, int c)
    {
       return a + b + c;
    }
};
int main()
{
```

```
    Test obj;
    cout<<"Sum of two integers "<<obj.sum(310, 220)<<endl;
    cout<<"Sum of three integers "<<obj.sum(12, 20, 23);
}
```

Output:

```
Sum of two integers 530
Sum of three integers 55
```

Try yourself:

(1) Write a program to swap two numbers using class

(2) Write a Program to Print Numbers From 1 to n using class

(3) Write a program to calculate area of a circle, a rectangle or a triangle depending on input using overloaded calculate function.

(4) Write a program that overloads the + operator and relational operators (suitable) to perform the following operations:

    a) Concatenation of two strings.

    b) Comparison of two strings.

9. Write a program to print factorial of a given number using recursive function.

```
//C++ Program to print factorial using recursive function
#include<iostream.h>
// Factorial Function
int factorial(int n)
{
    if (n > 1)
        return n * factorial(n - 1);  //recursive call of
factorial function
    else
        return 1;
}

int main()
{
    int n;
    cout << "Enter a number : ";
    cin >> n;
```
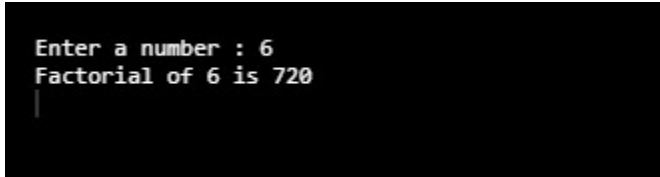
```
        cout << "Factorial of " << n << " is " << factorial(n);
        return 0;
}
```

Output:

```
Enter a number : 6
Factorial of 6 is 720
```

10. Write a program to print Fibonacci series using recursive function.

```cpp
//C++ Program to print Fibonacci series using recursive
function

#include<iostream.h>
int fibonacci (int n)
{
    if ((n==1)||(n==0))
     {
         return (n);
     }
     else
     {

 return (fibonacci (n-1)+fibonacci (n-2));  ////recursive
call of  fibonacci function
     }
}

int main()
{
     int n, i;

     cout<<"Enter number of terms for Fibonacci Series:";
      cin>>n;
     cout<<"Fibonacci Series "<<endl;

for (i=0; i< n; i++)
      {
```

```
        cout<<" "<<fibonacci (i);
    }
return 0;
}
```

Output:

```
Enter number of terms for Fibonacci Series:12
Fibonacci Series
 0 1 1 2 3 5 8 13 21 34 55 89
```

Try yourself:

(1) Write a program that uses a recursive function to find the binary equivalent of a given non-negative integer n.

(2) Write a programs functions to find the GCD of two given integers using recursive function.

Input/Output with files

C++ provides various header files for Input/output to/from data files. These header files are **ofstream**.h, **ifstream**.h and **fstream**.h.

- **ifstream**: Stream class to read from files.
- **ofstream**: Stream class to write on files.
- **fstream**: Stream class to both read and write from/to files.

11. Write a C++ program to create a file (data.txt).

```
/ /basic file operations
#include <iostream.h>
#include <fstream.h>
#include <conio.h>
void main ()
{
  ofstream file1;
  file1.open ("data.txt");
  file1 << "This is my first file.\n";
  file1.close ();
getch();
}
```

The above code will create a file called data.txt. We have inserted "This is my first file." text in our file.

Opening a file

```
open () function is used with filename and mode parameters
for opening a file.
```

Syntax:

```
open (filename, mode);
```

Filename parameter is used to give a filename to be opened and mode parameter is used to give modes in which file should be open. There are various modes available such as ios::in, ios::out, ios::binary, ios::ate etc. All these modes can be combined using bitwise OR (|) operator.

Example of combining file modes is:

```
ofstream file1;
file1.open ("data.bin", ios::out | ios::app | ios::binary);
```

Table 1.1 Modes of opening a file.

`ios::in`	This mode open file in input mode only.
`ios::out`	This mode open file in output mode only.
`ios::binary`	This mode open file in binary mode.
`ios::ate`	This mode set the initial position at the end of the file. If this flag is not set, the initial position is the beginning of the file.
`ios::app`	This mode allows all output operations are be performed at the end of the file (EoF), appending the content to the current content of the file.

Closing a file

```
Close () function is used for closing a file.
```

Syntax:

```
filename.close ();
```

12. Write a program for creating and writing on a text file.

```cpp
// C++ program of writing on a text file
#include<iostream.h>
#include<conio.h>
#include<fstream.h>
void main()
{
    ofstream file_out;
    char file_name[20];
    char str[80];
    clrscr ();
    cout<<"Enter file name to be created ";
    cin>> file_name;
```

```
    //create a new file in output mode
    file_out.open (file_name, ios::out);

    cout<<"Enter data to be stored ";
    cin>> str;

    file_out << str;
    cout<<"Information stored in file";

    //close file
    file_out.close ();
    getch();
}
```

13. Write a program to retrieve/read data from a text file.

```
// C++ program of retrieve data from a text file
#include<iostream.h>
#include<fstream.h>
#include<conio.h>

void main()
{
    ifstream file_in;
    char file_name[20];
    char str[80];

    clrscr ();

    cout<<"Enter file name: ";
    cin>> file_name;

    cout<<"Enter file name to open";
    file_in.open (file_name, ios::in);

    file_in.get (str, 80);
    cout<<str;

    file_in.close ();
    getch ();
}
```

Creating Binary file

File streams include two functions to perform read () and write () operations on a binary file.

```
write (void *buffer, int size);
read (void *buffer, int size);
```

The buffer is of type char* which represents the address of an array of bytes where the read data elements are stored. The size parameter is an integer value that specifies the number of characters to be read or written from/to the memory block.

14. Write a program for reading and writing on a binary file.

```cpp
//Read and Writing in a Binary File
#include<iostream.h>
#include<fstream.h>
#include<cstdio.h>

class Student
{
    int rno;
    char name[50];
public:
    void setData()
    {
        cout << "\n Enter roll number";
        cin >>rno;
        cout << "Enter name ";
        cin.getline (name, 50);
    }
    void showData ()
    {
        cout << "\nAdmission no. : " << rno;
        cout << "\nStudent Name : " << name;
    }

    };
// function to write in a binary file.
void write_data()
{
```

```
   ofstream file_out;
  file_out.open ("student.dat", ios::binary | ios::app);

   Student obj;
   obj.setData ();
  file_out.write((char*)&obj, sizeof(obj));
   file_out.close();
}


//function to display records of file

void display()
{
   ifstream file_in;
   file_in.open("student.dat", ios::binary);

   Student obj;

   while(file_in.read((char*)&obj, sizeof(obj)))
    {
        obj.showData();
    }

   file_in.close();
}


};


int main()
{
    for(int i = 1; i <= 4; i++)
      write_record ();

    //Display all records
    cout << "\nList of records";
    display ();

    //Search record
    cout << "\nSearch result";
    search (100);
```

```
                    //Delete record
                    delete_record(100);
                    cout << "\nRecord Deleted";
```

```
                    //Modify record
                    cout << "\nModify Record 101 ";
                    modify_record (101);
                    return 0;
             }
```

Try yourself

(1) What task does the following program perform?

```
       #include<iostream.h>
       #include<fstream.h>
       int main()
       {     ofstream ofile;
       ofile.open ("text.txt");
       ofile << "geeksforgeeks" << endl;
       cout << "Data written to file" << endl;    ofile.close();
       }
```

(2) Write a program which copies one file to another.

(3) Write a program to that counts the characters, lines and words in the text file.

---

15. Write a Program to print values of array using Pointers.

```
//C++ program of pointers with arrays
#include <iostream.h>
void main()
{
    int arr[5],i,*ptr;
    ptr = arr;   //ptr pointer is holding address of arr[0]
element.
        cout << "Enter 5 numbers:\n ";
            for(i=0;i<5;i++)
        cin >> arr[i];
    cout<<"\n Array values using pointer "<<endl;
      for(i=0;i<5;i++)
        cout<<*(ptr + i)<<endl;            //*(ptr+i) will
print array values

}
```

BLOCK III   LINEAR DATA STRUCTURE

1. Write a Program to implement Stack using Array.

```cpp
//C++ Program to implement Stack using Array
#include <iostream.h>
#define MAX 5
// stack class declaration
class stack
{
    private:
        int top;
        int ele[MAX];
    public:
//Default Constructor
stack()
{
    top = -1;
}
//PUSH function
void push( int item )
{
    if( top==MAX-1 )
    {
        cout<<"Stack Overflow"<<endl;
    }
else
{
    top++;
    ele[top] = item;

    cout<<"\n Inserted value is : "<< item;
}
}
 //POP function
int pop( )
{
    int item;
    if( top==-1 )
```

```
                  {
                      cout<<"\nStack Underflow";
                      }
 else
{
 item = ele[top];

top—;
}
return item;
 }

//DISPLAY function
void display ()
{
    if( top==-1 )
     {
         cout<<"\nStack Underflow";
         }
 else
{
 int i;
 cout<<"Stack value are "<<endl;

for(i=top;i>=0;i—)
cout<<ele[i]<<endl;
}
 }
}; //class closed

void  main()
{
int item = 0, choice, value; char ans;
    stack s = stack();
do
{
        cout<<"1.  Push"<<endl<<"2.  Pop"<<endl<<"3.
Display"<<endl<<"4. Exit";
    cout<<"\nEnter your choice: ";
    cin>>choice;
```

```
     switch(choice)
     {
      case 1:
        cout<<"Enter the value to be insert: ";
        cin>>value;
         s.push(value);
         break;
      case 2:
        value=s.pop();
         cout<<"\nDeleted value is "<<value;
         break;
      case 3:
        s.display();
         break;
      case 4:
     // exit(0);
      default:
        cout<<"Invalid choice";
     }

     cout<<"\nDo you want to cont…(y/n)";
     cin>>ans;
     }while (ans=='y' || ans=='Y');

  }
```

NOTES

Output:

```
1. Push"<<endl<<"2. Pop"<<endl<<"3. Display"<<endl<<"4. Exit
Enter your choice: 1
Enter the value to be insert: 11

 Inserted value is : 11
Do you want to cont…(y/n)y1. Push"<<endl<<"2. Pop"<<endl<<"3. Display"<<endl<<"4. Exit
Enter your choice:
1
Enter the value to be insert: 22

 Inserted value is : 22
Do you want to cont…(y/n)y
1. Push"<<endl<<"2. Pop"<<endl<<"3. Display"<<endl<<"4. Exit
Enter your choice: 3
Stack value are
22
11

Do you want to cont…(y/n)y1. Push"<<endl<<"2. Pop"<<endl<<"3. Display"<<endl<<"4. Exit
Enter your choice:
2

Deleted value is 22
Do you want to cont…(y/n)y
1. Push"<<endl<<"2. Pop"<<endl<<"3. Display"<<endl<<"4. Exit
Enter your choice: 3
Stack value are
11

Do you want to cont…(y/n)n
```

Algorithm: To transform infix expression to postfix expression value from stack using array.

Description: Here Q is an arithmetic expression written in Infix expression. Algorithm finds P as a Postfix expression

Stack is an array where we will PUSH and POP values.

MAX is a constant used to define maximum limit of TOP is the top most element of Stack where insertion and deletion is allowed.

DATA is data to be inserted in Stack.

1. Push "(" onto Stack, and add ")" to the end of Q.

2. Scan Q from left to right and repeat Step 3 to 6 for each element of Q until the Stack is empty.

3. If an operand is encountered, add it to P.

4. If a "(" left parenthesis is encountered THEN push it onto Stack.

5. If an operator is encountered ,then:

    1. Repeatedly POP from Stack and add to P each operator (on the top of Stack) which has the same precedence as or higher precedence than operator.

    2. Add operator to Stack.

        ```
        [End of If]
        ```

6. If a right parenthesis is encountered ,then:

    1. Repeatedly pop from Stack and add to P each operator (on the top of Stack) until a left parenthesis is encountered.

    2. Remove the left Parenthesis.

        ```
        [End of If]
        [End of If]
        ```

7. END.

2. Write a program to convert infix expression to postfix expression.

```
//C++ Program to convert INFIX EXPRESSION TO POSTFIX
EXPRESSION
#include<iostream.h>
# define MAX 100

//STACK CLASS DECLARATION
class stack
{
 public:
  char stack_array[MAX];
   int top;
```

```
//DEFAULT CONSTRUCTOR USED TO INITIALIZE TOP=-1
stack()
{
  top=-1;
}
//PUSH FUNCTION
void push(char symbol)
{
  if (full())
  {
   cout<<"\nStack overflow:\n";
  }
  else
  {
    top=top+1;
    stack_array [top] =  symbol;
  }
}
  //POP FUNCTION
 char pop()
 {
   if (empty())
    return ('#');        // Return value '#' indicates
stack is empty
   else
    return (stack_array[top-1);
 }
 int empty()
 {
   if (top==-1)
     return (1);
   else
     return (0);
 }
 int full()
 {
   if (top==49)
     return (1);
   else
     return (0);
 }
```

```
}; //STACK CLASS CLOSED
//EXPRESSION CLASS DECLARATION
class Expression
{
  char infix[MAX];
  char postfix[MAX];
  public:
  //INPUT FUNCTION TO TAKE INFIX EXPRESSION FROM THE USER
    void input()
    {
     cout<<"\nEnter an infix expression: (Ex. 2+3-(7*6)";
      cin>>infix;
    }
    int white_space(char symbol)
    {
     if(symbol==' ' || symbol=='\t' || symbol=='\0')
     return 1;
      else
     return 0;
    }
    //POSTFIX CONVERSION FUNCTION
 void ConvertToPostfix()
    {
      stack s;
      int l,precedence,p;
      char entry1,entry2;
      p=0;
     for(int i=0;infix[i]!='\0';i++)
      {
    entry1=infix[i];
    if(!white_space(entry1))
     {
       Switch (entry1)
       {
         case '(':
           s.push(entry1);
            break;
         case ')':
           while ((entry2=s.pop())!='(')
            postfix [p++]=entry2;
```

```
       break;
    case '+':
    case '-':
    case '*':
    case '/':
      if (!s.empty())
       {
   precedence = prec(entry1);
    entry2=s.pop();
   while (precedence<=prec(entry2))
    {
       Postfix [p++] =entry2;
       if (!s.empty())
          entry2= s.pop ();
        else
           break;
     }
    if (precedence>prec(entry2))
       s.push (entry2);
       }
      s.push (entry1);
     break;
      default:
       postfix [p++]=entry1;
        break;
     }
     }
   }
  while (!s.empty())              //while stack is not
empty
     postfix [p++]=s.pop();

 postfix [p]='\0';
 cout<<"\nThe postfix expression is: "<<postfix<<endl;
 }
 int prec(char symbol)
 {
  Switch (symbol)
   {
     case '/':
```

```
                    // Precedence of / is 4
                        return(4);
                case '*':
                    // Precedence of * is 3
                        return(3);
                case '+':
                    // Precedence of + is 2
                        return(2);
                case '-':
                     // Precedence of - is 1
                        return(1);
                case '(':
                    // Precedence of ( is 0
                        return(0);
                default:
                            return(-1);
            }
        }
    };
    int main()
    {
     char ch='y';
     Expression expr;
     do
     {
       expr.input();
       expr.ConvertToPostfix();
       cout<<"\n\nDo you want to continue ? (y/n): ";
       cin>>ch;
     }while(ch=='y' || ch=='Y');
     return 0;
    }
```

Output:


```
Enter an infix expression: (Ex. 2+3-(7*6)  4+5(2-3)

The postfix expression is: 4523-+


Do you want to continue ? (y/n):
```

Algorithm: For evaluation of postfix expression using stack.

Description:

Here P is an arithmetic expression written in postfix expression. Stack is an array where we will PUSH all the operands and final value.

TOP is the top most element of Stack where insertion and deletion is allowed

1. Add a Right Parenthesis ")" at the end of P postfix expression.

2. Scan P from left to right and repeat Step 3 to 4 for each element of P until ")" is encountered.

3. If an operand is encountered, add it to STACK.

4. If an operator * is encountered, then

   a) Remove two TOP elements of the stack where A is the TOP and B is the TOP-1 element.

   b) Evaluate B * A.

   c) Place the result of step (b) on to Stack

5. Set value equal to TOP element on stack.

6. Exit.

3. Write a program to evaluate postfix expressions using stack.

```cpp
//C++ program to evaluate of Postfix Expressions Using
Stack
#include <iostream.h>
#include <stdlib.h>
#include <math.h>
#include <ctype.h>
const int MAX = 50 ;
//Postfix class declaration
class postfix
{
    private:
        int stack[MAX] ;
        int top, nn ;
        char *s ;
    public:
//constructor
 postfix ( )
{
    top = -1 ;
}
```

```
void setexpr ( char *str )
{
    s = str;
}
void push ( int item )
{
    if ( top == MAX - 1 )
        cout << endl << "Stack overflow " ;
    else
    {
        top++;
        stack [top] = item ;
    }
}
int pop( )
{
    if ( top == -1 )
    {
        cout << endl << "Stack underflow " ;
        return NULL ;
    }
    int data = stack[top] ;
    top—;
    return data ;
}
void calculate( )
{
    int n1, n2, n3 ;
    while ( *s )
    {
        if ( *s == ' ' || *s == '\t' )
        {
            s++;
            continue;
        }
        if ( isdigit ( *s ) )
        {
            nn = *s – '0' ;
            push ( nn ) ;
        }
```

```
      else
      {
         n1 = pop ( );
         n2 = pop ( );
         switch ( *s )
         {
            case '+' :
               n3 = n2 + n1 ;
               break ;
            case '-' :
               n3 = n2 - n1 ;
               break ;
            case '/' :
               n3 = n2 / n1 ;
               break ;
            case '*' :
               n3 = n2 * n1 ;
               break ;
            case '%' :
               n3 = n2 % n1 ;
               break ;
            case '$' :
               n3 = pow ( n2 , n1 ) ;
               break ;
            default :
               cout << "Invalid operator" ;
               exit ( 1 ) ;
         }

         push ( n3 ) ;
      }
      s++ ;
   }
}
void show( )
{
   nn = pop ( ) ;
   cout << "Final Result : " << nn ;
}
```

```
} ;
void main( )
{
    char expr[MAX] ;
    cout << "\nEnter postfix expression to be evaluated :
" ;
    cin.getline (expr, MAX);
    postfix q ;
    q.setexpr (expr);
    q.calculate ( );
    q.show ( );
}
```

Algorithm to Push Value in Stack Using Linked List

Description:

```
Here TOP is a pointer variable which contains the address
of Top node.
NEW is the new node to be inserted in linked list.
INFO is data to be inserted in linked list
PUSH_STACK (TOP, INFO)
[OVERFLOW?]
1 IF NEW = NULL THEN
    WRITE:  OVERFLOW
    EXIT
2 ELSE
    IF   TOP   =   NULL      THEN
    SET   NEXT    [TOP]   =    NULL

    ELSE
    SET    NEXT    [NEW]    =    TOP
    [END OF IF]
    SET    DATA   [TOP]    = INFO
    SET   TOP   =   NEW
    [END OF IF]
3   END
```

Algorithm To Pop Value From Stack Using Linked List

Description:

Here TOP is the Top node of Stack to be deleted
TEMP is name given to the node to be deleted from the list

```
POP_STACK   (TOP)
[UNDERFLOW?]
 IF   TOP = NULL THEN
WRITE: UNDER FLOW
EXIT
ELSE
SET INFO=DATA [TOP]
SET TEMP = TOP
    SET TOP  =NEXT [TOP]
DELETE TEMP
    [END OF IF]
END
```

4. Write a program to implement stack using linked list.

```cpp
//C++ program for Stack implementation using linked list
#include <iostream.h>

//structure for NODE
struct NODE
{
   int data;
    NODE *next;
};

class STACK
{
private:
   NODE *top,*temp,*curr;
public:
   STACK ()
    {
        top= NULL;
    }
//PUSH Function to insert value to Stack
void push(int d)
{
     temp = new NODE; //allocate memory to new node
      temp->data = d;
 if (temp==NULL)
 {
    cout<<"Overflow";
```

```
}
else if (top==NULL)
{
    top=temp;
    top->next=NULL;

}
else
{
    temp->next=top;
    top=temp;

}


}
//POP Function to delete value from Stack
int pop()
{
    int d;
 if (top==NULL)
 {
    cout<<"Underflow";
}
else
{
    d= top->data ;
    temp=top;
    top=top->next;
    delete temp;
        }
   return d;
}

//Function to traversal/print STACK
   void traversal()
    {
        if(top == NULL)
        {
            cout << "Underflow" << endl;
        }
```

```
        else
        {
            curr= top;
            while (curr!=NULL)
            {cout << curr->data << endl;

            curr= curr->next;
        }

        }
    }


};


//MAIN FUNCTION
int main()
{
    STACK s;
    int d, ch;
    char ans;


  cout<<" Stack operations";

        do
    {
        cout<<"\n 1.Push\n 2.Pop\n 3.Print ";
        cout<<"\n Enter your choice \n";
        cin>>ch;
    switch (ch)
        {
    case 1:
            {
     cout<<"\nEnter value to be inserted ";
     cin>>d;
     s.push (d);
     break;
            }
    case 2:
```

```
                        {
                            d=s.pop ();
                             cout<<"\n Deleted value "<<d;
                              break;
                        }

            case 3:
                     {
                          cout<<"\n Stack Value are  \n";
                          s.traversal ();
                           break;
                     }
               default:
                     {
                           cout<<"\n Invalid choice  \n";
                     }
                 }
        cout<<"\n\nCont...(y/n)";
         cin>>ans;
          }
      while (ans=='y'|| ans=='Y');
          }
```

Output:

```
 Stack operations
 1.Push
 2.Pop
 3.Print
 Enter your choice 1

Enter value to be inserted 11


Cont...(y/n)y
 1.Push
 2.Pop
 3.Print
 Enter your choice
1

Enter value to be inserted 44


Cont...(y/n)y
 1.Push
 2.Pop
 3.Print
 Enter your choice
3

 Stack Value are
44
11
```

Try yourself:

(1) Write a program to solving Towers of Hanoi problem that using a recursive function.

(2) Write a program to convert infix expression to prefix expression.

Queues: Queue Implementation, Applications of Queue

Algorithms for queue Insertion using Array

```
INSERT _QUEUE (INT DATA)
IF (REAR=SIZE) THEN
WRITE "QUEUE IS FULL"
ELSE IF (REAR=0) THEN
FRONT=1
END IF
REAR=REAR+1
QUEUE [REAR]=DATA
EXIT
```

Algorithms for queue deletion using Array

```
DELETE-QUEUE ( )
IF (FRONT=0) THEN
WRITE "QUEUE IS EMPTY"
ELSE
ITEM=QUEUE [FRONT]
IF (FRONT=REAR) THEN
REAR=0
FRONT=0
ELSE
FRONT=FRONT+1
END IF
END IF
EXIT
```

5. Write a program to implement queue using linked list.

```cpp
//C++ program for Queue implementation using linked list
#include <iostream>
using namespace std;
//structure for NODE
struct NODE
{
    int data;
```

```
    NODE *next;
};

class QUEUE
{
private:
    NODE *front,*rear,*temp,*curr;
public:
    QUEUE()
      {
          front=rear= NULL;
      }
//enQueue (Insertion) Function to insert value to Queue
void enQueue(int d)
{
      temp = new NODE; //allocate memory to new node
        temp->data = d;
 if (temp==NULL)
 {
     cout<<"Overflow";
}

else if (front==NULL)
{
     front=rear=temp;
    rear->next=NULL;

}
else
{
    rear -> next = temp;
      rear = temp;

     }

}
//deQueue (delete) Function to delete value from Queue
int deQueue()
{
    int d;
```

```
  if(front == NULL)
 {
     cout<<"Queue is Empty";
}
else
{
    d= front->data;
    temp=front;
     front = front -> next;
    delete temp;
        }
   return d;
}


//Function to traversal/print STACK
   void traversal()
    {
        if (front == NULL)
        {
            cout << "Underflow" << endl;
        }
        else
        {
            curr=front;
            while (curr!=NULL)
            {cout << curr->data << endl;

            curr=curr->next;
        }

        }
    }


};


//MAIN FUNCTION
int main()
{
    QUEUE q;
```

```
 int d, ch;
 char ans;
cout<<" Queue operations";

    do
 {
    cout<<"\n 1.Insert \n 2.Delete \n 3.Print ";
    cout<<"\n Enter your choice \n";
    cin>>ch;
 switch(ch)
     {
         case 1:
         {
             cout<<"\nEnter value to be inserted ";
              cin>>d;
              q. enQueue( d);
             break;
         }
   case 2:
         {
            d=q.deQueue();
             cout<<"\n Deleted value "<<d;
             break;
         }

   case 3:
         {
             cout<<"\n Queue Value are  \n";
             q.traversal();
             break;
         }
    default:
         {
             cout<<"\n Invalid choice  \n";
         }
     }
 cout<<"\n\nCont...(y/n)";
 cin>>ans;
 }while(ans=='y'|| ans=='Y');
 }
```

Output:

```
Queue operations
 1.Insert
 2.Delete
 3.Print
 Enter your choice
1

Enter value to be inserted 55

Cont...(y/n)y
 1.Insert
 2.Delete
 3.Print
 Enter your choice
1

Enter value to be inserted 77

Cont...(y/n)y
 1.Insert
 2.Delete
 3.Print
 Enter your choice
3

 Queue Value are
55
77
```

Try yourself:

(1) Write a program to implement queue using array.

(2) Write a program to implement priority queue.

6. Write a program to insert a node at the beginning of singly linked list.

```cpp
//C++ program to insert node at the beginning of single
linked list
#include <iostream.h>

//structure for NODE
struct NODE
{
    int data;
    NODE *next;
};
```

```cpp
class linked_list
{
private:
    NODE *start,*end,*temp,*curr;
public:
    linked_list()
     {
         start = end= NULL;
     }
//Function to insert at the beginning
    void insert_beginning (int d)
     {
         temp = new NODE; //allocate memory to new node
         temp->data = d;
         temp->next = NULL;

         if(start == NULL)
          {
              start =end= temp;
              end->next=NULL;
          }
          else
          {
              temp->next = start;
              start=temp;
          }
     }
//Function to traversal/print single linked list
    void traversal()
     {
         if(start == NULL)
          {
              cout << "Underflow" << endl;
          }
          else
          {
              curr=start;
              while(curr!=NULL)
              {cout << curr->data << endl;
              curr=curr->next;
```

```
            }


            }
        }

};
void main()
{
    linked_list list;
     int d;
     char ans;
     do
     {
     cout<<"\nEnter value to be inserted ";
     cin>>d;
     list.insert_beginning (d);
     cout<<"\n\nCont...(y/n)";
     cin>>ans;
     }while(ans=='y'|| ans=='Y');

    cout<<"\n Value are \n";
     list.traversal();

}
```

Output:

```
Enter value to be inserted 11


Cont...(y/n)y
Enter value to be inserted 22


Cont...(y/n)y
Enter value to be inserted 33


Cont...(y/n)n
 Value are
33
22
11
```

7. Write a program to insert a node at the end of singly linked list.

```cpp
//C++ program to Insert Node at the end of single linked
list
#include <iostream.h>
struct NODE
{
    int data;
    NODE *next;
};
class linked_list
{
private:
   NODE *start,*end,*temp,*curr;
public:
    linked_list()
     {
        start = end= NULL;
     }
  //Function to insert at the end
    void insert_end(int d)
     {
        temp = new NODE; //allocate memory to new node
        temp->data = d;
        temp->next = NULL;

        if(start == NULL)
         {
            start =end= temp;
            end->next=NULL;
         }
        else
         {
            end->next=temp;
            end=temp;
         }
     }
    //Function to traversal/print single linked list
    void traversal()
     {
```
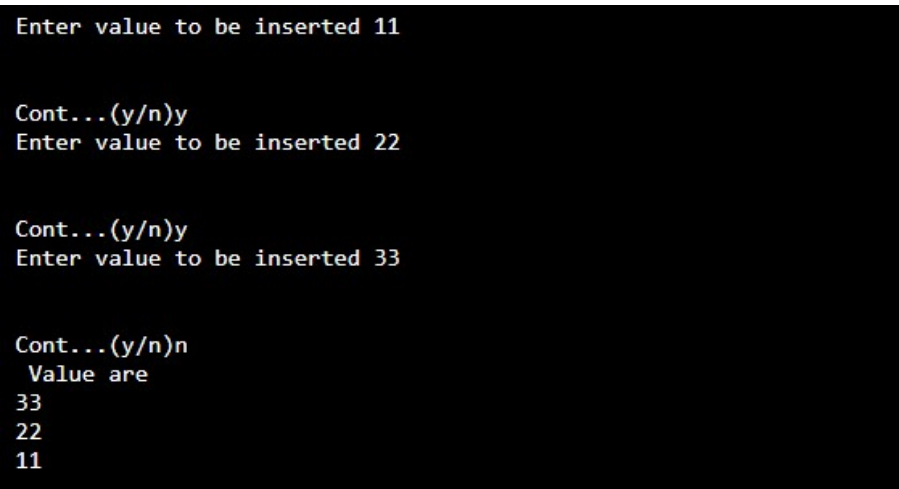
```
        if(start == NULL)
         {
             cout << "Underflow" << endl;
         }
         else
         {
             curr=start;
             while(curr!=NULL)
             {cout << curr->data << endl;

             curr=curr->next;
         }

         }
     }
};


void main()
{
       linked_list list;
    int d;
    char ans;
    do
    {
    cout<<"\nEnter value to be inserted ";
    cin>>d;
    list.insert_end(d);
    cout<<"\n\nCont...(y/n)";
    cin>>ans;
    }while(ans=='y'|| ans=='Y');

   cout<<"\n Value are \n";
    list.traversal();

}
```

Output:

```
Enter value to be inserted 11


Cont...(y/n)y
Enter value to be inserted 22


Cont...(y/n)y
Enter value to be inserted 33


Cont...(y/n)n
 Value are
11
22
33
```

8. Write a program to insert a node at the beginning of double linked list.

```cpp
//C++ program to insert node at the beginning of double
linked list
#include <iostream.h>

//structure for NODE
struct NODE
{
    NODE *prev;
    int data;
    NODE *next;
};
class linked_list
{
private:
    NODE *start,*end,*temp,*curr;
public:
    linked_list()
     {
        start = end= NULL;
     }
//Function to insert at the beginning in double linked
list
    void insert_beginning (int d)
     {
        temp = new NODE; //allocate memory to new node
        temp->data = d;
```

```
        temp->next = NULL;
        if(start == NULL)
         {
            start =end= temp;
            end->next=NULL;
            end->prev=NULL;
         }
        else
          {
         temp->prev=NULL;
         temp->next=start;
         start->prev=temp;
          start=temp;
         }
}


//Function to traversal/print double linked list (START
to END)
    void traversal_S_to_E()
     {
        if(start == NULL)
        {
            cout << "Underflow" << endl;
        }
        else
        {
            curr=start;
            while (curr!=NULL)
            {cout << curr->data << endl;

            curr=curr->next;
        }

        }
    }

//Function to traversal/print double linked list (END to
START)
    void traversal_E_to_S()
     {
        if(end == NULL)
```

```
        {
            cout << "Underflow" << endl;
        }
        else
        {
            curr=end;
            while(curr!=NULL)
            {cout << curr->data << endl;

            curr=curr->prev;
        }


        }
    }


};

//MAIN FUNCTION
int main()
{
    linked_list list;
    int d;
    char ans;
    do
    {
    cout<<"\nEnter value to be inserted ";
    cin>>d;
    list.insert_beginning (d);
    cout<<"\n\nCont...(y/n)";
    cin>>ans;
    }while(ans=='y'|| ans=='Y');

   cout<<"\n Value of list from start to end \n";
   list.traversal_S_to_E();
    cout<<"\n Value of list from end to start \n";
   list.traversal_E_to_S();

}
```

Output:

```
Enter value to be inserted 4


Cont...(y/n)y
Enter value to be inserted 34


Cont...(y/n)y
Enter value to be inserted 65


Cont...(y/n)n
 Value of list from start to end
65
34
4

 Value of list from end to start
4
34
65
```

9. Write a program to insert a node at the end of double linked list.

```cpp
//C++ program to insert node at the end of double linked
list
#include <iostream.h>
//structure for NODE
struct NODE
{
    NODE *prev;
    int data;
    NODE *next;
};
class linked_list
{
private:
    NODE *start,*end,*temp,*curr;
public:
    linked_list ()
    {
        start = end= NULL;
    }
//Function to insert at the end in double linked list
    void insert_end (int d)
    {
        temp = new NODE; //allocate memory to new node
        temp->data = d;
        temp->next = NULL;
        if(start == NULL)
        {
```

```
                    start =end= temp;
                    end->next=NULL;
                    end->prev=NULL;
                }
                else
                 {
                 end->next=temp;
                 temp->prev=end;
                 temp->next=NULL;
                  end=temp;
                 }
}
//Function to traversal/print double linked list (START
to END)
    void traversal_S_to_E()
     {
         if(start == NULL)
         {
             cout << "Underflow" << endl;
         }
         else
         {
             curr=start;
             while (curr!=NULL)
             {cout << curr->data << endl;
             curr=curr->next;
         }
         }
     }
//Function to traversal/print double linked list (END to
START)
    void traversal_E_to_S()
     {
         if (end == NULL)
         {
             cout << "Underflow" << endl;
         }
         else
         {
             curr=end;
             while (curr!=NULL)
             {cout << curr->data << endl;
             curr=curr->prev;
         }
```

```
        }
     }
};
//MAIN FUNCTION
int main()
{
    linked_list list;
    int d;
    char ans;
    do
    {
    cout<<"\nEnter value to be inserted ";
    cin>>d;
    list.insert_end (d);
    cout<<"\n\nCont...(y/n)";
    cin>>ans;
    }while (ans=='y'|| ans=='Y');
   cout<<"\n Value of list from start to end \n";
    list.traversal_S_to_E ();
     cout<<"\n Value of list from end to start \n";
    list.traversal_E_to_S ();
}
```

Output:

```
Enter value to be inserted 3


Cont...(y/n)y

Enter value to be inserted 43


Cont...(y/n)y
Enter value to be inserted 56


Cont...(y/n)n
 Value of list from start to end
3
43
56

 Value of list from end to start
56
43
3
```

Try yourself:

(1) Write a program to implement circular linked list.

(2) Write a program to merge two linked lists.

(3) Write a program to sort a linked list.

BLOCK IV   NON LINEAR DATA STRUCTURE

Algorithm: For inorder traversal

A binary tree t is in memory. This algorithm does an inorder traversal of t. An array stack is used to temporarily hold the address of nodes.

```
INORDER_TRAVERSAL (INFO, LEFT, RIGHT, ROOT)
1.  [PUSH NULL TO STACK AND INITIALIZE PTR]
    SET TOP = 1
    SET STACK [TOP] =NULL
    SET PTR = ROOT


2.  REPEAT WHILE PTR '" NULL
    SET TOP = TOP +1
    SET STACK [TOP] = PTR
    SET PTR = LEFT [PTR]
    [END OF LOOP]
3.  SET PTR = STACK[TOP]
    SET TOP = TOP -1
4.  REPEAT STEPS 5 TO 7 WHILE PTR '" NULL
5.  APPLY PROCESS TO INFO[PTR]
6.  [RIGHT CHILD?]
    IF RIGHT [PTR] '" NULL THEN
    SET PTR = RIGHT [PTR]
    GOTO STEP 2
    [END OF IF]
7.  SET PTR = STACK[TOP]
    SET TOP = TOP -1
    [END OF STEP 4 LOOP]
8.  END
```

Algorithm: For pre-order traversal

A binary tree t is in memory. This algorithm does a preorder traversal of t. An array stack is used to temporarily hold the address of nodes.

```
PREORDER _TRAVERSAL (INFO, LEFT, RIGHT, ROOT)
1.  [INITIALLY PUSH NULL TO STACK AND INITIALIZE PTR]
    SET TOP =1
    SET STACK [TOP]:=NULL
    SET PTR = ROOT
2.  REPEAT STEPS 3 TO 5 WHILE PTR '" NULL
```

```
3.  APPLY PROCESS TO INFO[PTR]
4.  [RIGHT CHILD?]
    IF RIGHT [PTR] '" NULL THEN
    SET PTR = RIGHT [PTR]
    SET TOP = TOP +1
    SET STACK [TOP]:= RIGHT [PTR]
    [END OF IF]

5.  [LEFT CHILD?]
    IF LEFT [PTR] '" NULL THEN
    SET PTR = LEFT [PTR]
    ELSE
    SET PTR = STACK [TOP]
    SET TOP = TOP -1
    [END OF IF]
    [END OF LOOP]
6.  END
```

Algorithm: For post-order traversal

A binary tree t is in memory. This algorithm does postorder traversal of t. An array stack is used to temporarily hold the address of nodes.

```
POSTORDER _TRAVERSAL (INFO, LEFT, RIGHT, ROOT)
1.  [INITIALLY PUSH NULL TO STACK AND INITIALIZE PTR]
    SET TOP =1
    SET STACK [1] =NULL
    SET PTR =ROOT
2.  [PUSH LEFT-MOST PATH ONTO STACK ]
    REPEAT STEPS 3 TO 5 WHILE PTR '" NULL
3.  SET TOP:=TOP+1
    SET STACK [TOP] =PTR

    [PUSHES PTR ON STACK]
4.  IF RIGHT[PTR] '" NULL THEN [PUSH ON STACK]
    SET TOP =TOP+1
    SET STACK [TOP] = RIGHT [PTR]
    [END OF IF STRUCTURE]
5.  SET PTR = LEFT[PTR] [UPDATE POINTER PTR]
    [END OF STEP 2 LOOP]
6.  [POP NODE FROM STACK]
    SET PTR = STACK [TOP]
    SET TOP =TOP-1
```

```
7.  REPEAT WHILE PTR>0
    a. APPLY PROCESS TO INFO[PTR]
    b. [POP NODE FROM STACK]
    SET PTR = STACK [TOP]
    SET TOP =TOP-1
    [END OF LOOP]
8.  IF PTR<0 THEN
    a) SET PTR:= -PRT
    b) GOTO STEP 2
    [END OF IF STRUCTURE]
9.  END
```

1. Write a program for binary tree insertion and in-order traversal.

```cpp
//C++ Program for inorder traversal of binary tree
# include <iostream.h>
//structure for NODE
struct NODE
{
    int data;
    struct NODE *left;
    struct NODE *right;
}*root;

class BST
{
    public:

        //constructor
        BST ()
        {
            root = NULL;
        }
 // Function to insert value into tree
void insert(NODE *tree, NODE *newnode)
{
    if (root == NULL)
    {
        root = new NODE;
        root->data = newnode->data;
        root->left = NULL;
        root->right = NULL;
```

```
        cout<<"Root Node is Added"<<endl;
         return;
   }
   if (tree->data == newnode->data)
    {
       cout<<"Element already in the tree"<<endl;
        return;
    }
   if (tree->data > newnode->data)
    {
        if (tree->left != NULL)
         {
            insert(tree->left, newnode);
         }
         else
         {
            tree->left = newnode;
            (tree->left)->left = NULL;
            (tree->left)->right = NULL;
            cout<<"Node Added To Left"<<endl;
             return;
         }
     }
     else
     {
         if (tree->right != NULL)
          {
             insert (tree->right, newnode);
          }
          else
          {
             tree->right = newnode;
             (tree->right)->left = NULL;
             (tree->right)->right = NULL;
             cout<<"Node Added To Right"<<endl;
              return;
          }
     }
}
```

```
        // Function for inorder tree traversal
void inorder (NODE *ptr)
{
    if (root == NULL)
    {
        cout<<"\n Underflow\n";
        return;
    }
    if (ptr != NULL)
    {
        inorder(ptr->left);
        cout<<ptr->data<<"  ";
        inorder(ptr->right);
    }
}


        // function to display tree structure
 void display(NODE *ptr, int level)
{
    int i;
    if (ptr != NULL)
    {
        display (ptr->right, level+1);
        cout<<endl;
        if (ptr == root)
            cout<<"Root->:  ";
        else
        {
            for (i = 0;i < level;i++)
                cout<<"        ";
    }
        cout<<ptr->data;
        display (ptr->left, level+1);
}}
};

 // main function
int main()
{
    int ch, num;
```

```
BST bst;
NODE *temp;
char ans;
do
{
   cout<<"\n1.Insert Element "<<endl;
   cout<<"2.Inorder Traversal"<<endl;
   cout<<"3.Display Tree Structure"<<endl;
    cout<<"4. Exit"<<endl;

    cout<<"Enter your choice : ";
    cin>>ch;
    switch (ch)
    {
    case 1:
        temp = new NODE;
        cout<<"Enter value to be inserted : ";
        cin>>temp->data;
        bst.insert (root, temp);
         break;
    case 2:
        cout<<"Inorder Traversal of BST:"<<endl;
        bst.inorder (root);
         break;
    case 3:
        cout<<"Display BST:"<<endl;
        bst.display (root,1);
         break;
    case 4:
         break;
    default:
        cout<<"Invalid choice"<<endl;
    }
   cout<<"\n\nCont...(y/n)";
cin>>ans;
}while (ans=='y'|| ans=='Y');

}
```

Output:

```
1.Insert Element
2.Inorder Traversal
3.Display Tree Structure
4. Exit
Enter your choice : 1
Enter value to be inserted : 65
Node Added To Right


Cont...(y/n)y
1.Insert Element
2.Inorder Traversal
3.Display Tree Structure
4. Exit
Enter your choice : 1
Enter value to be inserted : 45
Element already in the tree


Cont...(y/n)y
1.Insert Element
2.Inorder Traversal
3.Display Tree Structure
4. Exit
Enter your choice : 2
Inorder Traversal of BST:
45   65

Cont...(y/n)n
```

2. Write a program for binary tree insertion and preorder traversal.

```cpp
//C++ Program for preorder traversal of binary tree
# include <iostream.h>
//structure for NODE
struct NODE
{
    int data;
    struct NODE *left;
    struct NODE *right;
}*root;

class BST
{
    public:
        //constructor
         BST ()
         {
             root = NULL;
         }
```

```
// Function to inserting value into tree
void insert(NODE *tree, NODE *newnode)
{
    if (root == NULL)
    {
        root = new NODE;
        root->data = newnode->data;
        root->left = NULL;
        root->right = NULL;
        cout<<"Root Node is Added"<<endl;
        return;
    }
    if (tree->data == newnode->data)
    {
        cout<<"Element already in the tree"<<endl;
        return;
    }
    if (tree->data > newnode->data)
    {
        if (tree->left != NULL)
        {
            insert (tree->left, newnode);
        }
        else
        {
            tree->left = newnode;
            (tree->left)->left = NULL;
            (tree->left)->right = NULL;
            cout<<"Node Added To Left"<<endl;
            return;
        }
    }
    else
    {
        if (tree->right != NULL)
        {
            insert (tree->right, newnode);
        }
        else
        {
```

```
                tree->right = newnode;
                (tree->right)->left = NULL;
                (tree->right)->right = NULL;
               cout<<"Node Added To Right"<<endl;
                return;
            }
        }
}

  // Function for preorder tree traversal
void preorder(NODE *ptr)
{
    if (root == NULL)
    {
         cout<<"\nUnderflow\n";
    }
    if (ptr != NULL)
    {
        cout<<ptr->data<<"  ";
        preorder (ptr->left);
        preorder (ptr->right);
    }
}
    };

 // main function
int main()
{
    int ch, num;
    BST bst;
    NODE *temp;
    char ans;
    do
    {
       cout<<"\n1.Insert element "<<endl;
       cout<<"2.preorder traversal"<<endl;
        cout<<"3. Exit"<<endl;

        cout<<"Enter your choice : ";
        cin>>ch;
        switch(ch)
```

```
              {
           case 1:
                temp = new NODE;
                cout<<"Enter value to be inserted : ";
                cin>>temp->data;
                bst.insert(root, temp);
                 break;
           case 2:
                cout<<"preorder Traversal of BST:"<<endl;
                bst.preorder(root);
                 break;
           case 3:
                 break;
           default:
                cout<<"Invalid choice"<<endl;
            }
          cout<<"\n\nCont...(y/n)";
       cin>>ans;
       }while(ans=='y'|| ans=='Y');


   }
```

Output:

```
1.Insert element
2.preorder traversal
3. Exit
Enter your choice : 1
Enter value to be inserted : 4
Root Node is Added


Cont...(y/n)y
1.Insert element
2.preorder traversal
3. Exit
Enter your choice : 1
Enter value to be inserted : 66
Node Added To Right


Cont...(y/n)y
1.Insert element
2.preorder traversal
3. Exit
Enter your choice : 2
preorder Traversal of BST:
4   66

Cont...(y/n)y
1.Insert element
2.preorder traversal
3. Exit
Enter your choice : 3


Cont...(y/n)
```

3. Write a program for binary tree insertion and post-order traversal.

```cpp
//C++ Program for postorder traversal of binary tree
# include <iostream.h>
```

```cpp
//structure for NODE
struct NODE
{
    int data;
    struct NODE *left;
    struct NODE *right;
}*root;

class BST
{
    public:

        //constructor
        BST ()
        {
            root = NULL;
        }
// Function to insert value into tree
void insert(NODE *tree, NODE *newnode)
{
    if (root == NULL)
    {
        root = new NODE;
        root->data = newnode->data;
        root->left = NULL;
        root->right = NULL;
        cout<<"Root Node is Added"<<endl;
        return;
    }
    if (tree->data == newnode->data)
    {
        cout<<"Element already in the tree"<<endl;
        return;
    }
    if (tree->data > newnode->data)
    {
```

```
        if (tree->left != NULL)
         {
             insert(tree->left, newnode);
         }
         else
         {
             tree->left = newnode;
             (tree->left)->left = NULL;
             (tree->left)->right = NULL;
             cout<<"Node Added To Left"<<endl;
              return;
         }
     }
     else
     {
         if (tree->right != NULL)
         {
             insert (tree->right, newnode);
         }
         else
         {
             tree->right = newnode;
             (tree->right)->left = NULL;
             (tree->right)->right = NULL;
             cout<<"Node Added To Right"<<endl;
              return;
         }
     }
}

 // Function for postorder tree traversal

 void postorder(NODE *ptr)
{
    if (root == NULL)
    {
       cout<<"\n Underflow"<<endl;
        return;
    }
    if (ptr != NULL)
```

```
                          {
                              postorder (ptr->left);
                              postorder (ptr->right);
                              cout<<ptr->data<<"  ";
                          }
                      }
                          };

 // main function
int main()
{
    int ch, num;
    BST bst;
    NODE *temp;
    char ans;
    do
    {
        cout<<"\n1.Insert element "<<endl;
        cout<<"2.post-order traversal"<<endl;
         cout<<"3. Exit"<<endl;

         cout<<"Enter your choice : ";
         cin>>ch;
         switch (ch)
          {
         case 1:
              temp = new NODE;
             cout<<"Enter value to be inserted : ";
              cin>>temp->data;
             bst.insert(root, temp);
              break;
         case 2:
             cout<<"post-order Traversal of BST:"<<endl;
             bst.postorder(root);
              break;
         case 3:
              break;
         default:
             cout<<"Invalid choice"<<endl;
          }
```

```
        cout<<"\n\nCont...(y/n)";
    cin>>ans;
    }while (ans=='y'|| ans=='Y');
  }
```

Output:

```
1.Insert element
2.post-order traversal
3. Exit
Enter your choice : 1
Enter value to be inserted : 22
Root Node is Added


Cont...(y/n)Y
1.Insert element
2.post-order traversal
3. Exit
Enter your choice : 3


Cont...(y/n)Y
1.Insert element
2.post-order traversal
3. Exit
Enter your choice : 2
post-order Traversal of BST:
22

Cont...(y/n)Y
1.Insert element
2.post-order traversal
3. Exit
Enter your choice : 1
Enter value to be inserted : 66
Node Added To Right

Cont...(y/n)Y
1.Insert element
2.post-order traversal
3. Exit
Enter your choice : 2
post-order Traversal of BST:
66  22

Cont...(y/n)
```

Try yourself:

(1) Write a program to check whether a tree is a binary search tree.

(2) Write a program to search an element in a tree recursively.

(3) Write a program for depth first binary tree search using recursion.

(4) Write a program to find the largest value in a tree using inorder traversal.

Graphs

Directed Graph (digraph)

In a *directed graph*, the edges are represented by *ordered pairs* of vertices (*u, v*) and shown diagrammatically as directed arrows.
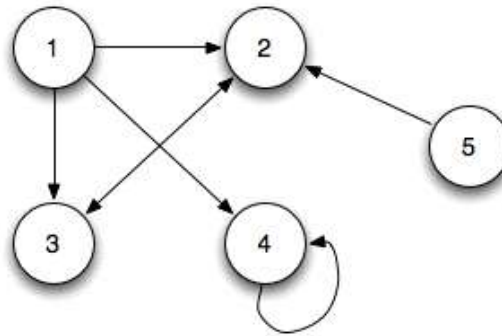
An edge (u, v) is *incident from* (i.e. leaves) u and is *incident to* (i.e. enters) v. If a graph contains an edge (u, v), then v is *adjacent* to u and is represented notationally as $u \to v$.

Note v being adjacent to u *does not* imply that u is adjacent to v unless edge $(v, u) \in E$. Thus (u, v) and (v, u) are distinct edges in a directed graph.

We say that u and v are *neighbours* if either $(u, v) \in E$ or $(v, u) \in E$.

For each vertex we define the *out-degree* as the number of edges leaving the vertex, *in-degree* as the number of edges entering a vertex, and the *degree* as *out-degree* + *in-degree* (i.e. total number of edges at the vertex). If a vertex has *degree* = 0, then the vertex is *isolated*.

If the directed graph has no self-loops, then it is a *simple directed graph*.

Undirected Graph

In an *undirected graph*, the edges are represented by *unordered pairs* of vertices. Thus (u, v) and (v, u) represent the same edge and are shown diagrammatically as simply a connecting line (note undirected graphs may not contain self-loops).
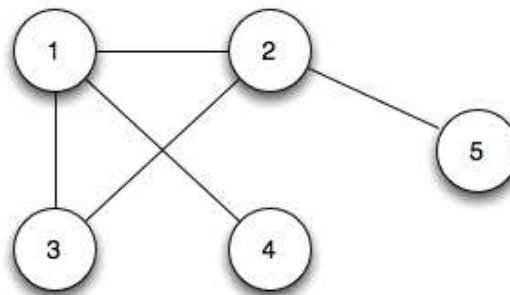
An edge (u, v) is *incident on* u and v, and u and v are *adjacent* to each other.

The *degree* is the number of edges incident on a vertex.

To convert an undirected graph into a directed one, simply replace each edge (u, v) with (u, v) and (v, u). Conversely to convert a directed graph into an undirected one, replace each edge (u, v) or (v, u) with (u, v) and remove all self-loops.

Paths

A *path* of length *k* from *u* to *u'* is a sequence of vertices $<v_0, v_1, ..., v_k>$ with $u = v_0$, $u' = v_k$, and $(v_{i-1}, v_i) \in E$.

If a path *p* from *u* to *u'* exists, then *u'* is *reachable* from *u* (denoted $u \rightarrow u'$ if *G* is a directed graph).

The path is *simple* if all the vertices are distinct.

A *subpath* is a contiguous subsequence $<v_i, v_{i+1}, ..., v_j>$ with $0 \leq i \leq j \leq k$.

A *cycle* is a path with $v_0 = v_k$ (and is also simple if all the vertices except the end points are distinct). An *acyclic graph* is a graph with no cycles.

Connected Components

In an undirected graph, a *connected component* is a subset of vertices thar are all reachable from each other. The graph is *connected* if it contains exactly one connected component, i.e. *every* vertex is reachable from *every* other.

In a directed graph, a *strongly connected component* is a subset of *mutually* reachable vertices, i.e. there is a path between any two vertices in the set.

Special Graphs

A *complete* graph an undirected graph where *all* vertices are adjacent to all other vertices, i.e. there are edges between every pair of vertices.

A *bipartite* graph is an undirected graph that can be *partitioned* into V_1 and V_2 such that for every edge $(u, v) \in E$ either

$u \in V_1$ and $v \in V_2$ **OR** $u \in V_2$ and $v \in V_1$

i.e. the graph can be separated so that the only edges are between vertices in different subsets.

A *forest* is an undirected acyclic graph. If it is also connected, then it is a *tree*.

A directed acyclic graph is known as a *DAG*.

Graph Representation

Two common methods for implementing a graph in software is either using an *adjacency list* or an *adjacency matrix*.

Adjacency List

In an adjacency list implementation, we simply store the adjacent vertices (i.e. edges) for each vertex in a linked list denoted **Adj**[*u*]. If we sum up the lengths of all the adjacency lists, we get either

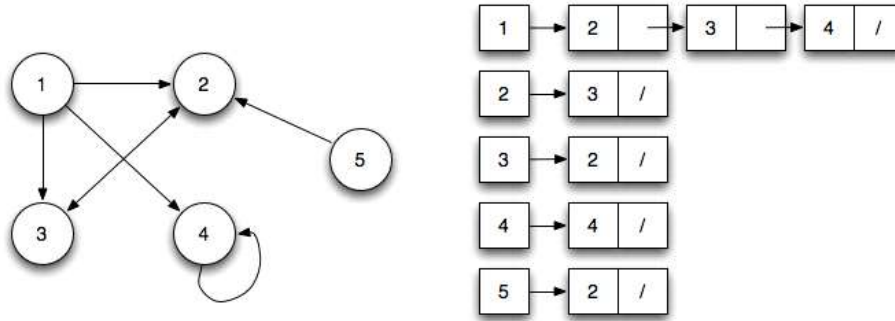$$\sum |\text{Adj}[u]| = \begin{cases} |E| & \text{if directed} \\ 2|E| & \text{if undirected} \end{cases}$$

$\Rightarrow \Theta(V + E)$ storage is required.

This representation is good for sparse graphs where $|E| << |V|^2$. One drawback is that to determine if an edge $(u, v) \in E$ requires a list search $\Rightarrow \Theta(V)$.

For the original directed graph, the adjacency list would be

Adjacency Matrix

In an adjacency matrix implementation, we store the edges in a V×V matrix A either as binary values or real numbers for weighted edges.

$\Rightarrow \Theta(V^2)$ storage is required (independent of E).

This representation is good for dense graphs where $|E| \approx |V|^2$. The advantage is it only takes $\Theta(1)$ time to determine if an edge $(u, v) \in E$ since it is simply a matrix element access. If the graph is undirected, then $A = A^T$ so only the upper triangular half needs to be stored.

For the original directed graph, the adjacency matrix would be

	1	2	3	4	5
1	0	1	1	1	0
2	0	0	1	0	0
3	0	1	0	0	0
4	0	0	0	1	0
5	0	1	0	0	0
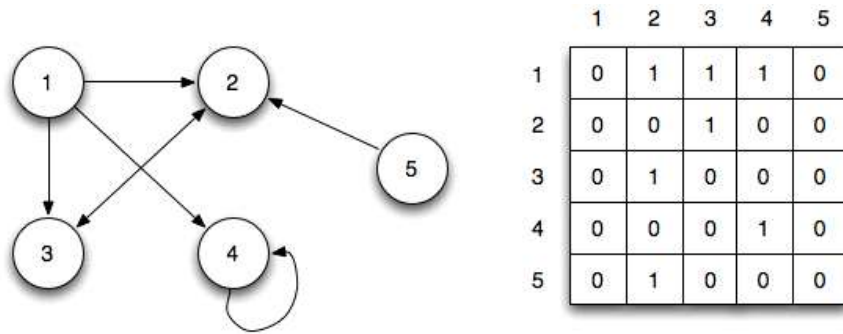
Shortest Path Algorithms

(i) Dijkstra's Algorithm

Dijkstra's algorithm maintains a set S of vertices where minimum paths have been found and a priority queue Q of the remaining vertices under discovery ordered by increasing u.d's.

DIJKSTRA (G,w,s)

1. INITIALIZE-SINGLE-SOURCE (G,s)
2. $S = \Theta$
3. Q = G.V
4. while $Q \neq \Theta$
5. u = EXTRACT-MIN(Q)
6. $S = S \cup \{u\}$
7. for each vertex $v \in G.Adj[u]$
8. RELAX(u,v,w)

INITIALIZE-SINGLE-SOURCE (G,s)

1. for each vertex $v \infty G.V$
2. $v.d = \infty$
3. v.pi = NIL
4. s.d = 0

RELAX (u,v,w)

1. if $v.d > u.d + w(u,v)$
2. $v.d = u.d + w(u,v)$
3. v.pi = u

(ii) Floyd-Warshall Algorithm for finding Shortest Path

The Floyd-Warshall algorithm works based on a property of intermediate vertices of a shortest path. An intermediate vertex for a path $p = <v_1, v_2, ..., v_j>$ is any vertex other than v_1 or v_j.

Algorithm:

FLOYD-WARSHALL(W)

1. n = W.rows
2. $D^{(0)} = W$
3. $\Pi^{(0)} = \pi^{(0)}_{ij} = $ NIL if i=j or $w_{ij} = \infty$
   $= i$ if $i \neq j$ and $w_{ij} < \infty$
4. for k = 1 to n
5. let $D^{(k)} = (d^{(k)}_{ij})$ be a new nxn matrix
6. for i = 1 to n
7. for j = 1 to n

8. $dk_{ij} = \min(d^{(k-1)}_{ij}, d^{(k-1)}_{ik} + d^{(k-1)}_{kj})$

9. if $d^{(k-1)}_{ij} \leq d^{(k-1)}_{ik} + d^{(k-1)}_{kj}$

10. $\pi^{(k)}_{ij} = \pi^{(k-1)}_{ij}$

11. else

12. $\pi^{(k)}_{ij} = \pi^{(k-1)}_{kj}$

13. return $D^{(n)}$

Basically the algorithm works by repeatedly exploring paths between every pair using each vertex as an intermediate vertex.

Minimum cost Spanning Trees

(i) Kruskal's Algorithm

Kruskal's algorithm is a greedy algorithm. To find out minimum spanning tree (MST) of a connected and weighted graph, we use Kruskal's algorithm. This means it finds a subset of the edges to forms a tree including each vertex in such a way that the total weight of all the edges in the tree is minimal. In case a graph is not a connected graph it finds a MST connected component and that is a minimum spanning forest.

Algorithm:

MST-KRUSKAL (G,w)

$A = \Theta$

for each vertex $v \in G.V$

MAKE-SET (v)

sort the edges of G.E into nondecreasing order by weight w

for each edge $(u,v) \in G.E$, taken in nondecreasing order by weight

if FIND-SET(u) $\neq$ FIND-SET(v)

$A = A \cup \{(u,v)\}$

UNION (u,v)

return A

Basically the algorithm works as follows:

1. Make each vertex a separate tree

2. Sort the edges in nondecreasing order

3. Add an edge if it connects different trees and merge the trees together

The run time of Kruskal's algorithm depends on the implementation of the set operations, but can be made to run in O(E lg V).

(ii) Prims Algorithm

Prim's algorithm is a very simple modification to Dijkstra's shortest path algorithm. With Prim's algorithm, you build the minimum spanning tree node by node. You

are going to maintain a "current spanning tree", which will be a subset of the nodes in the graph, and the edges that compose a minimum spanning tree of those nodes.

Algorithm:

MST-PRIM (G,w,r)

1. for each $u \in G.V$
2. $u.key = \infty$
3. $u.pi = NIL$
4. $r.key = 0$
5. $Q = G.V$
6. while $Q \neq \Theta$
7. $u = EXTRACT\text{-}MIN(Q)$
8. for each $v \in G.Adj[u]$
9. if $v \in Q$ and $w(u,v) < v.key$
10. $v.pi = u$
11. $v.key = w(u,v)$

Basically the algorithm works as follows:

1. Initialize Q and set the source (root) key to 0
2. While Q is not empty, dequeue the vertex with minimum weight edge and add it to the tree by adding edge $(u.\in,u)$ to T
3. For each vertex v in Adj[u] that is still in Q, check if w(u,v) (the edge weights from u for all vertices not in T) are less than the current v.key (the current smallest edge weight) and if so update the predecessor and key fields

Graph traversing Methods

1. Breadth First Search(BFS)
2. Depth First Search(DFS)

Algorithm for BFS (Breadth First Search):

This Algorithm executes a breadth first search on a graph G beginning at a starting node A

1. [Initialize all nodes to the ready state]

   Set status:=1
2. Put the starting node A in Queue and change its status to the

   Set status :=2
3. Repeat steps 4 and 5 until Queue is Empty:

4. Remove the front node N of Queue process N and change the status of N to the processed

state

Set status: =3

5. Add to the rear of Queue all the neighbors of N that are in the ready state (state=1), and

Change their status to the waiting state

Set status:=2

[End of step 3 loop]

End

Algorithm for DFS (Depth First Search)

This algorithm executes a Depth First Search on a graph G beginning A

1. [initialize all nodes to the ready state]

Set status: =1

2. Push the starting node A onto stack and change its status to the waiting state

Set status: =2

3. Repeat step 4 and 5 until stack is empty

4. Pop the top node N of stack . process N and change its status to the processed state

Set status: =3

5. Push onto stack all the neighbors of N that are still in the ready state (status=1),and change their status to the waiting state

Set status: =2

[End of step 3 loop]

6. End

DFS Applications:

DFS can be used directly, but is more often used as an intermediate technique within another algorithm. This lecture examines three applications of DFS.

- Parenthesis Theorem
- Topological Sorting
- Strongly Connected Component Decomposition (SCCD)

BLOCK V SEARCHING AND SORTING ALGORITHMS

Searching refers to the operation of finding the location of a given item in a collection of items.

Algorithm for Sequential Search

```
    INPUT            : LIST OF SIZE N, TARGET VALUE T
    OUTPUT           : POSITION OF T IN THE LIST
1.   BEGIN
2.  SET FOUND = FALSE
     SET  I = 0
3. WHILE    I≤N   AND FOUND IS FALSE
   IF LIST [I] = T THEN
    SET FOUND = TRUE
    EXIT
     ELSE
     SET I =I+1
     [END OF STEP 3 LOOP]
4. IF     FOUND = FALSE THEN
    WRITE: T IS NOT IN LIST
     ELSE
      WRITE: T IS FOUND AT I LOCATION
        [END OF IF]
5 .  END
```
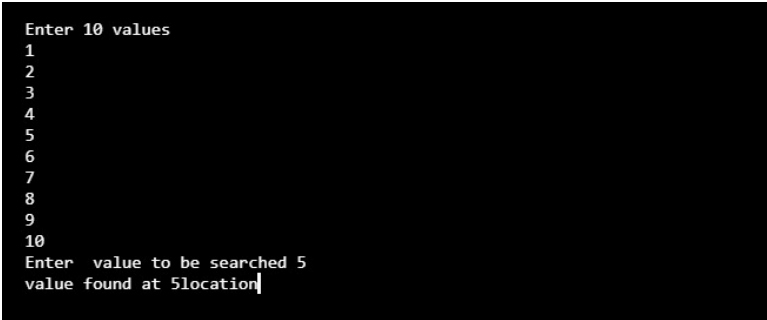
1. Write a program to search a given value in an array using sequential search.

```cpp
//C++ program for sequential search
#include <iostream.h>
//definition of sequential_Search function
void  sequential_search (int  a[ ] ,int size ,int key)
{
     int  flag  , i ;
    flag =0;
   for (  i=0  ;  i<size ;  i++)
   {
     if ( a [i] ==  key )
      {
       flag = 1 ;
       break ;
```

```
            }
      }
      if ( flag == 1)
      cout<<"value found at "<<i+1<<"location";
      else
     cout<<"value not found";
}
void main()
{
int arr[10],i,k;
cout<<"Enter 10 values";
for(i=0;i<10;i++)
cin>>arr[i];

cout<<"Enter  values to be searched";
cin>>k;
//call of sequential_search function
sequential_search (arr, 10, k);
}
```

Output:

Algorithm for Binary Search

```
    INPUT          : SORTED LIST OF SIZE N, KEY VALUE KEY
    OUTPUT         : POSITION OF KEY IN THE LIST = KEY
1. BEGIN
2. [INTIALIZE]
     SET MAX = SIZE
      SET MIN = 1
      SET FOUND = FALSE
 3. WHILE (FOUND IS FALSE AND MAX ?MIN)
          SET MID = (MAX + MIN)/2
4. IF  KEY = LIST [MID] THEN
          SET I = MID
```

```
            SET FOUND = TRUE
                 EXIT
         ELSE IF KEY < LIST [MID] THEN
                 SET MAX = MID -1
         ELSE
                      SET MIN = MIN +1
         [END OF IF]
         [END OF LOOP]
    5. IF FOUND = FALSE THEN
            WRITE: VALUE IS NOT IN LIST
         ELSE
        WRITE VALUE FOUND AT MID LOCATION
    6. END
```

2. Write a program to search an element in the array using binary search.

```cpp
//C++ program for binary Search

#include <iostream.h>
// Binary Search Function
void  binary_search (int a[ ] , int size , int key)
{
int low ,high ,mid ,flag ;
flag= 0;
low  = 0;
high = size -1;
while (low <= high && flag ==0)
{
mid =(low +high)/2;
if ( key == a [mid])
{
flag=1;
break;
}
else   if (key   < a[mid ] )
{
high  =  mid -1;
}
else
{
low  =  mid +1;
}
```

```
}
if  ( flag ==1)
{
cout<<"value found at  location"<<mid +1;
}
else
cout<<"value not found";
}


void main()
{
int arr[10],i,k;
cout<<"Enter 10 values\n";
for(i=0;i<10;i++)
cin>>arr[i];

cout<<"Enter  value to be searched ";
cin>>k;
//call of binary_Search function
binary_search(arr,10,k);
}
```

Output:

```
Enter 10 values
10
20
30
40
50
60
70
80
90
100
Enter  value to be searched 60
value found at  location6
```

Sorting techniques: Bubble sort, Quick sort, Insertion sort, Merge sort

Sorting refers to the operation of arranging data in some given order such as increasing or decreasing with numerical data and alphabetically with character data.

Selection Sort

Selection sort algorithm starts by comparing first two elements of an array and swapping if necessary.

This algorithm is not suitable for large data sets as its average and worst case complexities are of $\Theta(n^2)$, where n is the number of items.

3. Write a program to sort an array using selection sort.

```cpp
//C++ program for selection sort
#include <iostream.h>
void selection_sort (int a[ ],  int  size )
{
  int temp ,i,j, min;

        for(int i = 0; i < size-1 ; i++)  {

            min = i ; //considering element i as minimum

             for(int j = i+1; j < size ; j++ )
             {
                if(a[ j ] < a[ min ])
                 {
                 min = j ;
                 }
              }
                 temp= a[ min ];
           a[ min ]=a[ i ] ;
           a[ i]=temp;

        }
}
//main function
void main()
{
int arr[10],i;
cout<<"Enter 10 values\n";
for(i=0;i<10;i++)
cin>>arr[i];
//call of selection sort function
selection_sort(arr,10);
cout<<" \n Sorted Values \n";
for(i=0;i<10;i++)
cout<<endl<<arr[i];
}
```

Output:

4. Write a program to sort an array using bubble sort.

```
//C++ program for bubble sort
#include <iostream.h>
void bubble_sort (int a[ ],  int  size )
{
  int temp ,i,j;
    for (i=0; i<size; i++)
   {
     for (j=0; j<size-1; j++)
      {

          if(a[j]>a[j+1])
        {
         temp =a[j];
         a[j]=a[j+1];
         a[j+1]=temp;
        }      }
    }
}
/main function

void main()
{
int arr[10],i;
cout<<"Enter 10 values\n";
for (i=0;i<10;i++)
cin>>arr[i];
```

```
//call of bubble sort function
bubble_sort(arr,10);


cout<<" \n Sorted Values \n";
for(i=0;i<10;i++)
cout<<endl<<arr[i];
}
```

Output:

```
Enter 10 values
65
7
4
3
44
23
4
56
9
7

 Sorted Values

3
4
4
7
7
9
23
44
56
65
```

Insertion Sort Algorithm

```
   ALGORITHM    - INSERTION SORT
   INPUT        -  LIST [ ] OF     N    ITEMS
   OUTPUT      - LIST [ ] OF   N   ITEMS IN SORTED ORDER
1. BEGIN
2. FOR I = 2 TO N DO
3. IF   LIST [I] < LIST [I -1] THEN
4. SET I :  =  I  -  1
5. SET TEMP: = LIST [I]
6.  REPEAT
7. SET   LIST [J + 1]   = LIST [J];
8.  SET   I  : =   J - 1
9.  UNTIL   (J ≥ 1 AND LIST [J]> TEMP)
10. SET LIST [J+1]   : =TEMP
[END   OF   IF]
[END OF STEP 2 LOOP]
10. END
```

5. Write a C++ program to sort an array using insertion sort.

```
//C++ program for insertion sort
#include <iostream.h>
```

```
/insertion sort function definition
void insert_sort(int a[ ],int size)
{
int i,temp,j;
for (i = 1; i < size; i++)
    {
        temp = a[i];
        j = i-1;
while (j >= 0 && a[j] > temp)
        {
            a[j+1] = a[j];
            j--;
        }
        a[j+1] = temp;
}
}


//main function
void main()
{
int arr[10],i,k;
cout<<"Enter 10 values\n";
for(i=0;i<5;i++)
cin>>arr[i];
//call of Insertion Sort function
insert_sort (arr, 5);
cout<<" \n Sorted Values \n";
for (i=0;i<5;i++)
cout<<endl<<arr[i];
}
```
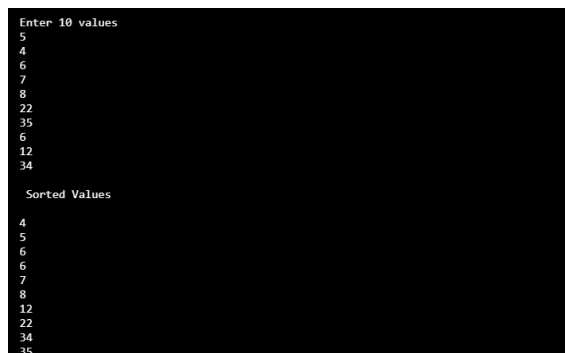
Output:

```
Enter 10 values
5
4
6
7
8
22
35
6
12
34

 Sorted Values

4
5
6
6
7
8
12
22
34
35
```

Algorithm for Quick Sort

```
QUICK_SORT ( ARRAY ,FIRST ,LAST )
1.  SET  LOW  :  =  FIRST
    SET   HIGH  :  =  LAST
    SET  PIVOT : =ARRAY[(   LOW +  HIGH) /2 ]
2.  REPEAT  THROUGH  STEP 7 WHILE (LOW  ?HIGH)
3. REPEAT STEP 4 WHILE (ARRAY  [LOW]<PIVOT)
4. SET LOW := LOW+1
5. REPEAT STEP 6 WHILE (ARRAY [HIGH]>PIVOT)
6. SET HIGH :=  HIGH-1
7. IF (LOW <=HIGH)
    ARRAY [LOW]   <->ARRAY [HIGH]
    SET LOW :=  LOW+1
     SET HIGH:=  HIGH-1
8. IF (FIORST<HIGH)  THEN
    QUICK_SORT (ARRAY,FIRST,HIGH)
9.  IF  (LOW < LAST)
     QUICK_SORT (ARRAY,LOW,LAST)
10.  END
```

6. Write a C++ program to sort an array using quick sort.

```cpp
//C++ program for quick sort
#include <iostream.h>
void quick_sort (int a[ ], int first, int last)
{
int   low ,high ,pivot, temp, i ;
low= first ;
high =last ;
pivot =a[(first +last)/2];
do
{
   while (a[low]<pivot)
    {
    low++;
    }
   while (a [high]>pivot)
    {
    high--;
    }
 if(low <=high)
  {
```
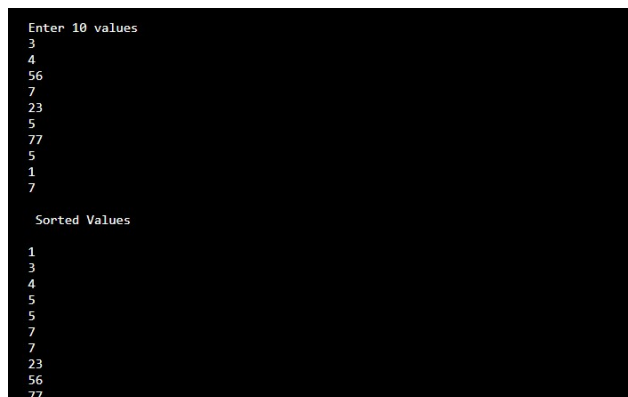
```
temp= a [low];
a [low]= a[high];
a[high]= temp ;
low++;
high--;
}
}while (low <=high);
if (first <high)
{
quick_sort (a, first, high);
}
if(low< last)
{
quick_sort (a, low, last);
}
}
void main()
{
int arr[10],i,k;
cout<<"Enter 10 values\n";
for(i=0;i<10;i++)
cin>>arr[i];
//call of Quick Sort function
quick_sort(arr,0,10);

cout<<" \n Sorted Values  \n";
for(i=0;i<10;i++)
cout<<endl<<arr[i];
}
```

Output:

Algorithm for Shell Sort

```
    ALGORITHM  _   SHELL  SHORT
    INPUT      _    LIST OF N ELEMENTS
    OUTPUT     _    LIST OF N ELEMENTS  IN ASSENDING ORDER
    SHELL_ SORT ( LIST  ,SIZE)
1. [INITIALIZE]
  SET  GAP  := N/2
2. REPEAT THROUGH STEP 6 WHILE GAP =0
      SET   SWAP  := 0
4. REPEAT THROUGH  STEP 6 WHILE SWAP=1
5. REPEAT THROUGH STEP 6  FOR I=1,3 ,.....I<(N-GAP)
6. IF ( LIST [I] > LIST [I+ GAP]) THEN
  SET  LIST [I] ,-.  LIST [I+ GAP]
   SET   SWAP  := 1
  [END OF FOR LOOP ]
  [END OF  INNER  WHILE  LOOP]
   [END  OF  OUTER  WHILE LOOP]
7. END
```

7. Write a program to implement shell sort.

```cpp
//C++ program for shell sort
#include <iostream.h>

void shell_sort (int a[ ],  int  size )


{
  int  temp  , gap  ,i ,swap ;
gap = size /2 ;
do
{
do
{
swap =0;
for ( i=0 ; i < size-gap ;  i ++)
{
if(a[i] > a[ i+ gap])
{
temp  =  a[i] ;
a[i]  = a  [i+ gap];
a[ i+ gap]= temp;
```
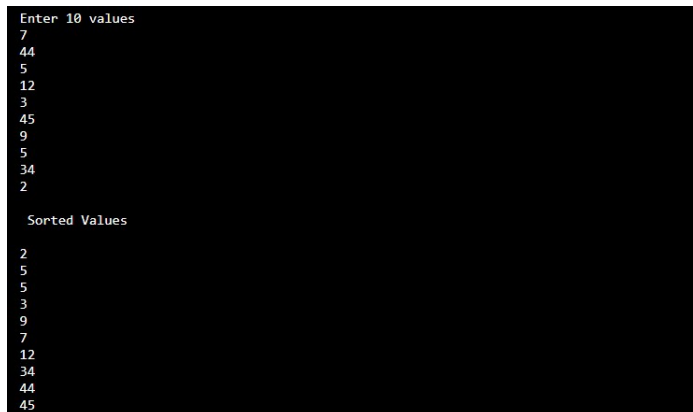
```
swap=1;
}}
}while  ( swap ==1);
gap= gap/2 ;
}while (gap >0) ;
}

void main()
{
int arr[10],i,k;
cout<<"Enter 10 values\n";
for(i=0;i<10;i++)
cin>>arr[i];
//call of shell sort function
shell_sort(arr,10);
cout<<" \n Sorted Values  \n";
for(i=0;i<10;i++)
cout<<endl<<arr[i];
}
```

Output:

```
Enter 10 values
7
44
5
12
3
45
9
5
34
2

 Sorted Values

2
5
5
3
9
7
12
34
44
45
```

8. Write a program to implement merge sort.

```
//C++ program for merge sort
#include <iostream.h>
// function to merge the two half into a sorted data.
void merge_array(int a[], int low, int high, int mid)
{
    // low to mid and mid+1 to high array are already
sorted
```

```
    int i, j, k;
    int temp_arr[high-low+1];
    i = low;
    k = 0;
    j = mid + 1;
    while (i <= mid && j <= high) // merging of two parts
into temp array
    {
        if (a[i] < a[j])
        {
            temp_arr[k] = a[i];
            k++;
            i++;
        }
        else
        {
            temp_arr[k] = a[j];
            k++;
            j++;
        }
    }
    while (i <= mid)    // insertion of remaining values
from i to mid into temp array.
    {
        temp_arr[k] = a[i];
        k++;
        i++;
    }
    while (j <= high) // insertion of remaining values
from j to high into temp array.
    {
        temp_arr[k] = a[j];
        k++;
        j++;
    }
    // assign sorted data stored in temp array to an array
    for (i = low; i <= high; i++)
    {
        a[i] = temp_arr[i-low];
    }
}
```
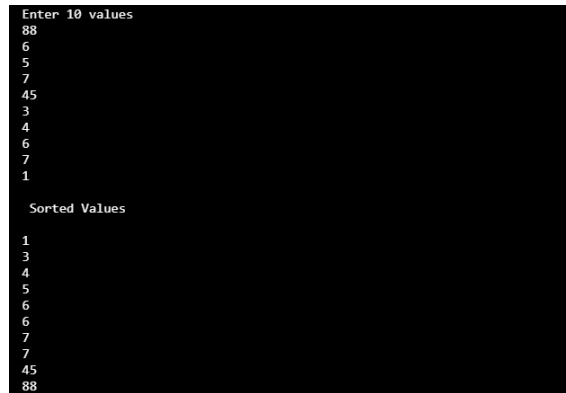
```
// A function to split array into two parts.
void merge_sort(int a[], int low, int high)
{
    int mid;
    if (low < high)
    {
        mid=(low+high)/2;
        // split array into two parts
        merge_sort(a, low, mid);
        merge_sort(a, mid+1, high);
        // merge arraythem to get sorted values
        merge_array(a, low, high, mid);
    }
}
 void main()
{
int arr[10],i,k;
cout<<"Enter 10 values\n";
for(i=0;i<10;i++)
cin>>arr[i];
//call of merge sort function
    merge_sort(arr, 0, 9);
cout<<" \n Sorted Values  \n";
for(i=0;i<10;i++)
cout<<endl<<arr[i];
}
```

Output:

Try yourself:

(1) Write a program to sort n numbers in descending order using bubble sort.

(2) Write a program to implement selection sort method using functions.

(3) Write a program to sort the n names in an alphabetical order.

Master of Computer Applications
315 14
LAB: DATA STRUCTURE USING C++

I - Semester

ALAGAPPA UNIVERSITY