STES's

SINHGAD COLLEGE OF ENGINEERING
Vadgaon(Bk), Pune

Department of Computer Engineering

LABORATORY MANUAL

2022-23

OBJECT ORIENTED PROGRAMMING & COMPUTER GRAPHICS LABORATORY
SE-COMPUTER ENGINEERING

SEMESTER-I

Subject Code:**210247**

TEACHING SCHEME   EXAMINATION SCHEME
Lectures: 4Hrs/Week   Practical :25 Marks
Practical: 2 Hrs/Week   Term Work:25 Marks

-: **Name of Faculty**:-
Prof. N. G. Bhojne
Prof. A. M. Karanjkar
Prof. P. D. Bendale
Prof. P. G. Waware

INDEX

GROUP C

Sr.No.	Title	Page Number
6	Write C++ program using STL for sorting and searching user defined records such as personal records (Name, DOB, Telephone number etc) using vector container. **OR** Write C++ program using STL for sorting and searching user defined records such as Item records (Item code, name, cost, quantity etc) using vector container.	50
7	Write a program in C++ to use map associative container. The keys will be the names of states and the values will be the populations of the states. When the program runs, the user is prompted to type the name of a state. The program then looks in the map, using the state name as an index and returns the population of the state.	57

PART-II : COMPUTER GRAPHICS

Suggested List of Laboratory Experiments/Assignments
(**All assignments are compulsory**)

GROUP A

Sr.No.	Title	Page Number
1	Write C++ program to draw a concave polygon and fill it with desired color using scan fill algorithm. Apply the concept of inheritance.	63
2	Write C++ program to implement Cohen Southerland line clipping algorithm.	68
3	A) Write C++ program to draw the following pattern. Use DDA line and Bresenham's circle drawing algorithm. Apply the concept of encapsulation. **OR** B) Write C++ program to draw the following pattern. Use DDA line and Bresenham's circle drawing algorithm. Apply the concept of encapsulation.	72

GROUP B

Sr.No.	Title	Page Number
4	a) Write C++ program to draw 2-D object and perform following basic transformations, Scaling b) Translation c) Rotation. Apply the concept of operator overloading. **OR** b) Write C++ program to implement translation, rotation and scaling transformations on equilateral triangle and rhombus. Apply the concept of operator overloading.	78
5	a) Write C++ program to generate snowflake using concept of fractals. **OR** b) Write C++ program to generate Hilbert curve using concept of fractals. **OR** c) Write C++ program to generate fractal patterns by using Koch curves.	85

GROUP C

6	a) Design and simulate any data structure like stack or queue visualization using graphics. Simulation should include all operations performed on designed data structure. Implement the same using OpenGL. **OR** b) Write C++ program to draw 3-D cube and perform following transformations on it using OpenGL i) Scaling ii) Translation iii) Rotation about an axis (X/Y/Z). **OR** c) Write OpenGL program to draw Sun Rise and Sunset.	92
7	a) Write a C++ program to control a ball using arrow keys. Apply the concept of polymorphism. **OR** b) Write a C++ program to implement bouncing ball using sine wave form. Apply the concept of polymorphism. **OR** c) Write C++ program to draw man walking in the rain with an umbrella. Apply the concept of polymorphism. **OR** Write a C++ program to implement the game of 8 puzzle. Apply the concept of polymorphism. **OR** d) Write a C++ program to implement the game Tic Tac Toe. Apply the concept of polymorphism	101

MINI PROJECT/CASE STUDY

8	Design and implement game / animation clip / Graphics Editor using open source graphics library. Make use of maximum features of Object Oriented Programming.	**106**

PART-I OBJECT ORIENTED PROGRAMMING

Suggested List of Laboratory Experiments/Assignments

(**All assignments are compulsory**)

GROUP A

Assignment No.	1 (GROUP A)
Title	Implement a class Complex which represents the Complex Number data type. Implement the following 1. Constructor (including a default constructor which creates the complex number 0+0i). 2. Overload operator+ to add two complex numbers. 3. Overload operator* to multiply two complex numbers. 4. Overload operators << and >> to print and read Complex Numbers
Subject	Object Oriented Programming
Class	S.E. (C.E.)
Roll No.	
Date	
Signature	

Assignment No: 1

Title: Arithmetic operations on complex numbers using operator overloading

Problem: Implement a class Complex which represents the Complex Number data type. Implement the following operations:

Statement:

1. Constructor (including a default constructor which creates the complex number 0+0i).
2. Overloaded **operator+** to add two complex numbers.
3. Overloaded **operator*** to multiply two complex numbers.
4. **Overloaded << and >>to print and read Complex Numbers**

Prerequisites: Object Oriented Programming

Objectives: To learn the concept of constructor, default constructor, operator overloading using member function and friend function.

Theory:

Operator Overloading : It is a specific case of polymorphism where different operators have different implementations depending on their arguments. In C++ the overloading principle applies not only to functions, but to operators too. That is, of operators can be extended to work not just with built-in types but also classes. A programmer can provide his or her own operator to a class by overloading the built-in operator to perform some specific computation when the operator is used on objects of that class.

An Example of Operator Overloading

Complex a(1.2,1.3); *//this class is used to represent complex numbers*

Complex b(2.1,3); *//notice the construction taking 2 parameters for the real and imaginary part*

Complex c = a+b; *//for this to work the addition operator must be overloaded*

Arithmetic Operators

Arithmetic Operators are used to do basic arithmetic operations like addition, subtraction, multiplication, division, and modulus. The following table list the arithmetic operators used in C++

Operator	Action
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus

With C++ feature to overload operators, we can design classes able to perform

operations usingstandard operators. Here is a list of all the operators that can be overloaded:

Over loadable operators
+ - * / = <> += -= *= /= <<>>
<<= >>= == != <= >= ++ -- % & ^ ! |
~ &= ^= |= && || %= []

- To overload an operator in order to use it with classes we declare *operator functions*, which are regular functions whose names are the operator keyword followed by the operator signthat we want to overload. The format is:
- type operator operator-symbol (parameters) {/*...*/ }

- The **operator** keyword declares a function specifying what *operator-symbol* means when applied to instances of a class. This gives the operator more than one meaning, or "overloads" it. The compiler distinguishes between the different meanings of an operator by examiningthe types of its operands.

Syntax:

return_typeclass_name :: operator op(arg_list)

{

//function body

}

where,

- Return type is the value returned by the specified operation
- op is the operator to be overload.
- op is proceeding by the keyword operator.
- operator op is the function name

Process of the overloading has 3 steps

1. Create a class that define a data types that is used in the overloading operation
2. Declare the operator function operator op() in the public part of the class.It may be either a member function or a friend function.
3. Define the operator function to implement the required operation

e.g. Overloading Binary operators: A statement like

C = sum (A, B);   // functional notation

This functional notation can be replaced by a natural looking expression
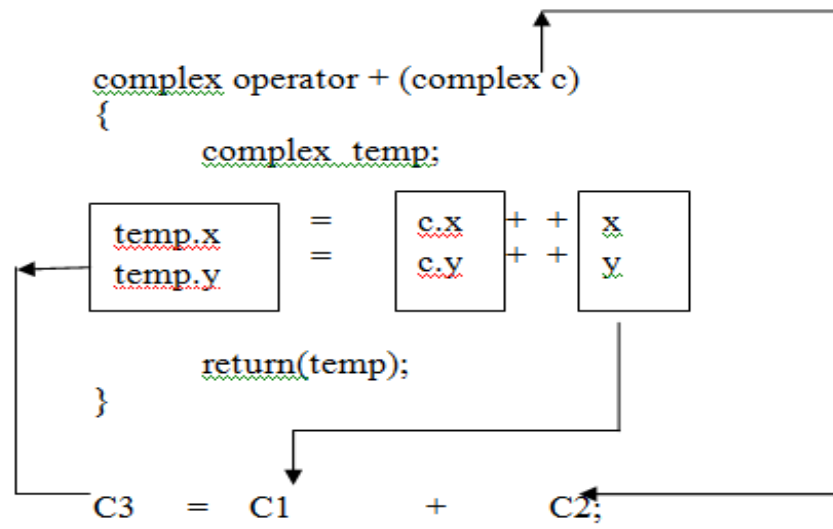
Facilities:

Linux Operating Systems, G++

Algorithm:
Step 1: Start the program

Step 2: Create a class complex

Step 3: Define the default constructor.

Step 4: Declare the operator function which are going to be overloaded and display function

Step 5: Define the overloaded functions such as +, -,/,* and the display function

For Addition:

(a+bi) + (x + yi) = ((a+x)+(b+y)i)

For Multiplication:
(a+bi) * (x + yi) = (((a*x)-(b*y)) + ((a*y) + (x*b))i)
Step 6: Create objects for complex class in main() function

Step 7:Create a menu for addition, multiplication of complex numbers and display the result

Step 8: Depending upon the choice from the user the arithmetic operators will invoke the overloaded operator automatically and returns the result

Step 9: Display the result using display function

Input:

Complex numbers with real and imaginary values for two complex numbers.

Example :

Complex No 1:	Real Part :	5
	Imaginary part :	4
Complex No 2:	Real Part :	5
	Imaginary part :	4

Output:

Default constructor value=0+0i
Enter the 1st number
Enter the real part2
Enter the imaginary part4
Enter the 2nd number
Enter the real part4
Enter the imaginary part8
The first number is 2+4i
The second number is 4+8i
The addition is 6+12i

The multiplication is -24+32i

Conclusion:

Hence, we have studied concept of operator overloading.

Questions:
1. What is operator overloading?
2. What are the rules for overloading the operators?
3. State clearly which operators are overloaded and which operator are not overloaded?
4. State the need for overloading the operators.
5. Explain how the operators are overloaded using the friend function.
6. What is the difference between "overloading" and "overriding"?
7. What is operator function? Describe the syntax?
8. When is Friend function compulsory? Give an example?

Assignment No.	2 (GROUP A)
Title	Personnel information system using constructor, destructor, static member functions, friend class, this pointer, inline code and dynamic memory allocation
Subject	Object Oriented Programming
Class	S.E. (C.E.)
Roll No.	
Date	
Signature	

Assignment No: 2

Title: Personnel information system using constructor, destructor, static member functions, friend class, this pointer, inline codeand dynamic memory allocation.

Problem: Develop an object oriented program in C++ to create a database of the personnel

Statement: information system containing the following information: Name, Date of Birth, Blood group, Height, Weight, Insurance Policy, number, Contact address, telephone number, driving license no. etc Construct the database with suitable member functions for initializing and destroying the data viz constructor, default constructor, copy, destructor, static member functions, friend class, this pointer, inline code and dynamic memory allocation operators-new and delete as well as exception handling.

Prerequisites: Object Oriented Programming

Objectives: To learn the concept of constructor, default constructor, copy, destructor, static member functions, friend class, this pointer, inline code and dynamic memory allocation operators-new and delete.

Theory:

Constructor:
A special method of the class that will be automatically invoked when an instance of the class is created is called as constructor. Following are the most useful features of constructor.
1) Constructor is used for Initializing the values to the data members of the Class.
2) Constructor is that whose name is same as name of class.
3) Constructor gets Automatically called when an object of class is created.
4) Constructors never have a Return Type even void.
5) Constructor is of Default, Parameterized and Copy Constructors.
The various types of Constructor are as follows: -
Constructors can be classified into 3 types
1. Default Constructor
2. Parameterized Constructor
3. Copy Constructor

1. **Default Constructor: -** Default Constructor is also called as Empty Constructor which has no arguments and It is Automatically called when we create the object of class but Remember name of Constructor is same as name of class and Constructor never declared with the help of Return Type. Means we can't declare a Constructor with the help of void Return Type., if we never Pass or declare any Arguments then this called as the Copy Constructors.

2. P**arameterized Constructor: -** This is another type constructor which has some Arguments and same name as class name but it uses some Arguments So For this, we have to create object of Class by passing some Arguments at the time of creating object with the name of class. When we pass some Arguments to the Constructor then this will automatically pass the Arguments to the Constructor and the values will retrieve by the Respective Data Members of the Class.

3. **Copy Constructor: -** This is also another type of Constructor. In this Constructor we pass the object of class into the Another Object of Same Class. As name Suggests you Copy, means Copy the values of one Object into the another Object of Class .This is used for Copying the values of class object into an another object of class So we call them as Copy Constructor and For Copying the values We have to pass the name of object whose values we wants to Copying and When we are using or passing an Object to a Constructor then we must have to use the & Ampersand or Address Operator.

Destructor: As we know that Constructor is that which is used for Assigning Some Values to data Members and For Assigning Some Values this May also used Some Memory so that to free up the Memory which is Allocated by Constructor, destructor is used which gets Automatically Called at the End of Program and we doesn't have to Explicitly Call a Destructor and Destructor Can't be Parameterized or a Copy This can be only one Means Default Destructor which Have no Arguments. For Declaring a Destructor, we have to use ~tiled Symbol in front of Destructor.

Static members
A class can contain static members, either data or functions.
A static member variable has following properties:
• It is initialized to zero when the first object of its class is created. No other initialization is permitted.
• Only one copy of that member is created for the entire class and is shared by all the objects of that class.
• It is the visible only within the class but its lifetime is the entire program

Static data members of a class are also known as "class variables", because there is only one unique value for all the objects of that same class. Their content is not different from one object static members have the same properties as global variables but they enjoy class scope. For that reason, and to avoid them to be declared several times, we can only include the prototype (its declaration) in the class declaration but not its definition (its initialization). In order to initialize a static data-member we must include a formal definition outside the class, in the global scope of this class to another. Because it is a unique variable value for all the objects of the same class, it can be referred to as a member of any object of that class or even directly by the class name (of course this is only valid for static members.

A static member function has following properties

• A static function can have access to only other static members (fun or var) declared in the same class

• A static function can be called using the class name instead of its object name
  Class_name :: fun_name;

Static member functions are considered to have class scope. In contrast to non static member functions, these functions have no implicit this argument; therefore, they can use only static data members, enumerators, or nested types directly. Static member functions can be accessed without using an object of the corresponding class type.

The following restrictions apply to such static functions:

1. They cannot access non static class member data using the member-selection operators (. or –>).

2. They cannot be declared as virtual.

3. They cannot have the same name as a non-static function that has the same argument types.

E.g. // static members in classes

```
class StaticTest {

 private: static int x;

 public: static int

 count()

 {

   return x;

     }

 };
 intStaticTest::x = 9;
```

```
int main()
 {
printf_s("%d\n", StaticTest::count());
 }
```

Output:

9

Friend functions:

In principle, private and protected members of a class cannot be accessed from outside the same class in which they are declared. However, this rule does not affect friends. Friends are functions or classes declared as such. If we want to declare an external function as friend of a class, thus allowing this function to have access to the private and protected members of this class, we do it by declaring a prototype of this external function within the class, and preceding it with the keyword friend.

Properties of friend function:

• It is not in the scope of the class to which it has been declared as friend.

• Since it is not in the scope of the class , it cannot be called using the object of that class

• It can be invoked like a normal function w/o the help of any object.

• It can be declared in private or in the public part of the class.

• Unlike member functions, it cannot access the member names directly and has to use an object name and dot operator with each member name.

```
// friend function
#include<iostream>
Using namespace std;
 Class  CRectangle
 {
 int width, height;
 public:
        void set_values (int, int);
     int area () { return (width * height);
     }
       friend CRectangle duplicate (CRectangle);
     };
```

```
    void CRectangle::set_values(int a, int b)
    {
    width = a;
    height=b;
    }
    CRectangle duplicate (CRectanglerectparam)
    {
    CRectanglerectres;
    rectres.width= rectparam.width*2;
    rectres.height= rectparam.height*2;
    return (rectres);
    }
    int main ()
    {
    CRectanglerect, rectb;
    rect.set_values (2,3);
    rectb = duplicate (rect);
    cout<<rectb.area();
    return 0;
    }
```

The duplicate function is a friend of CRectangle. From within that function we have been able to access the members width and height of different objects of type CRectangle, which are private members. Notice that neither in the declaration of duplicate() nor in its later use in main() have we considered duplicate a member of class CRectangle.

Friend classes

Just as we have the possibility to define a friend function, we can also define a class as friend ofanother one, granting that second class access to the protected and private members of the first one.

```
    // friend class.
#include <iostream>
using namespace std;
class CSquare;
class CRectangle

 {  int width, height;

public:
     int area ()
```

```
   {
    return (width * height);
     }
   void convert(Csquare a);
   };
class CSquare {
private:
int inside;
public:
void set_side(int a)
{ side=a; }
friend class CRectangle;
};
void CRectangle ::convert (CSquare a)
{
width=a.side;
height=a.side;
}
 void main ()
     { CSquaresqr;
     CRectanglerect;
     sqr.set_side(4);
     rect.convert(sqr);
     cout<<rect.area();
     }
```

In this example, we have declared CRectangle as a friend of CSquare so that CRectangle member functions could have access to the protected and private members of CSquare, more concretely to CSquare::side, which describes the side width of the square..

Pointers:

A pointer is a derived data type that refers to another data variable by storing the variables memory address rather than data.

Declaration of pointer variable is in the following form :

```
Data_type * ptr_var;
```

Eg int *ptr;

Here ptr is a pointer variable and points to an integer data type.

We can initialize pointer variable as

followsint a, *ptr; // declaration

ptr = &a //initialization

Pointers to objects:

Consider the following eg item X;// where item is class and X is object Similarly we can define a pointer it_ptr of type item as followsItem *it_ptr ;

Object pointers are useful in creating objects at runtime. We can also access public

members of theclass using pointers.

Eg   item X;

item *ptr = &X;

the pointer 'ptr 'is initialized with address of X.

we can access the member functions and data using pointers as

followsptr->getdata();

ptr->show();

this pointer:

C++ uses a unique keyword called this to represent an object that invokes a member function.

this is a pointer that points to the object for which this function was called. This unique pointer

is automatically passed to a member function when it is called.

Important notes on this pointer:

- this pointer stores the address of the class instance, to enable pointer access of the members tothe member functions of the class.

- this pointer is not counted for calculating the size of the object.
- this pointers are not accessible for static member functions. this pointers are not modifiable.

Facilities:

Linux Operating Systems, G++

Algorithm:

1. Start
2. Read personnel information such as Name, Date of Birth, Blood group, Height, Weight,Insurance Policy, number, Contact address, telephone number, driving license no..
3. Print all information from database.
4. Stop

Input:
Personnel information such as Name, Date of Birth, Blood group, Height, Weight, Insurance Policy, number, contact address, telephone number, driving license no.
Output:
Display personnel information from database. The result in following format:

    Name   DOB   ……. Driving License No

    1
    2

        .
        .

.   N

Conclusion:
Hence, we have successfully studied concept of constructor, default constructor, copy constructor,destructor, static member functions, friend class, this pointer, inline code and dynamic memory allocation operators-new and delete.

Questions:

1. What is static Function?
2. What is friend function? State the advantage of using the friend function.
3. What is friend class? Explain with examples.

4. Explain with examples pointers to object.
5. What is this pointer? Explain with examples.
6. State the advantages of this pointer.
7. What are inline functions?
8. How do we declare member of a class static?
9. What are demerits of friend function?
10. What is concept of constructor, destructor? What are types of constructors?

Assignment No.	3 (GROUP A)
Title	Creating a class which uses the concept of inheritance, displays data and data members and uses the concept of exception handling.
Subject	Object Oriented Programming
Class	S.E. (C.E.)
Roll No.	
Date	
Signature	

Assignment No: 3

Title: Creating a class which uses the concept of inheritance, displays data and data members and uses the concept of exception handling.

Problem Statement: Imagine a publishing company which does marketing for book and audio cassette versions. Create a class publication that stores the title (a string) and price (type float) of publications. From this class derive two classes: book which adds a page count (type int) and tape which adds a playing time in minutes (type float). Write a program that instantiates the book and tape class, allows user to enter data and displays the data members. If an exception is caught, replace all the data member
values with zero values.

Prerequisites: Object Oriented Programming

Objectives: To learn the concept of inheritance and exception handling.

Theory:

Inheritance:
Inheritance in Object Oriented Programming can be described as a process of creating new classes from existing classes. New classes inherit some of the properties and behavior of the existing classes. An existing class that is "parent" of a new class is called a base class. New class that inherits properties of the base class is called a derived class. Inheritance is a technique of code reuse. It also provides possibility to extend existing classes by creating derived classes.

The basic syntax of inheritance is:

Class DerivedClass : accessSpecifier BaseClass

There are 3 access specifiers: Namely **public, private and protected**.
public: This inheritance mode is used mostly. In this the protected member of Base class becomes protected members of Derived class and public becomes public.
protected: In protected mode, the public and protected members of Base class becomes protected members of Derived class.

private: In private mode the public and protected members of Base class become private members of Derived class.
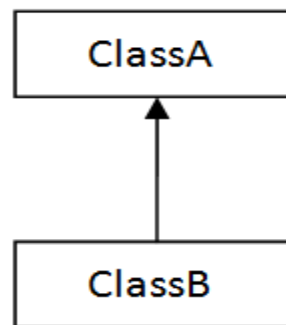
Types of Inheritance

In C++, we have 5 different types of Inheritance. Namely,

1. Single Inheritance
2. Multiple Inheritance
3. Hierarchical Inheritance
4. Multilevel Inheritance
5. Hybrid Inheritance

Single Inheritance:

In this type of inheritance one derived class inherits from only one base class. It is the most simplest form of Inheritance.

Single Inheritance.

Syntax:

```
class subclass_name : access_modebase_class
{
//body of subclass
};
```

using namespace std;

class Vehicle
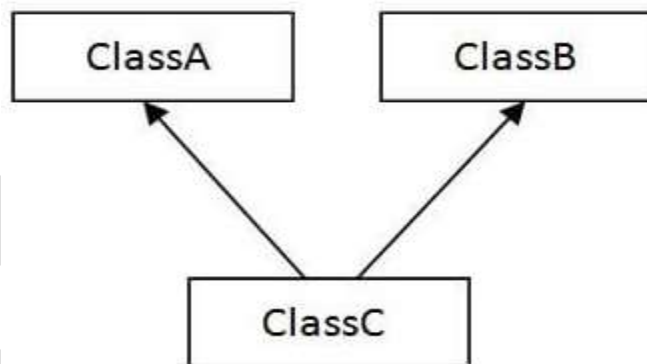 {

public:

      Vehicle()
      { cout<< "This is a Vehicle"<<endl;
      }
    };
    Class Car: publicVehicle
    {
    };
   int main()
   {
   Car obj;
   return 0;
   }

Output:

This is a vehicle

Multiple Inheritance:

In this type of inheritance, a single derived class may inherit from two or more than two base classes.

Multiple Inheritance

Syntax:

classsubclass_name : access_mode base_class1, access_mode base_class2, ...

{

//body of subclass
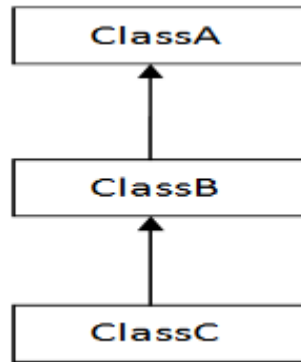
};

// Multiple Inheritence

```cpp
#include<iostream>
using namespace std;
class Vehicle
{
 public:
 Vehicle()
 {
 cout<< "This is a Vehicle"<<endl;
 }
};
class FourWheeler
{public:
FourWheeler()
{
cout<< "This is a 4 wheeler Vehicle"<<endl;
 }
};
class Car: public Vehicle, public FourWheeler
 {
};

int main( ) {

Car obj;
return 0;
}
```

Output:
 This is a Vehicle
 This is a 4 wheeler Vehicle

Multilevel Inheritance: In this type of inheritance the derived class inherits from a class, which in turn inherits from some other class. The Super class for one, is sub class for the other.

Multilevel Inheritance

```cpp
#include <iostream>
using namespace std;
class Animal {
  public:
void eat() {
  cout<<"Eating..."<<endl;
}
  };
  class Dog: public Animal
  {
    public:
   void bark(){
   cout<<"Barking..."<<endl;
    }
  };
  class BabyDog: public Dog
  {
    public:
   void weep( ) {
   cout<<"Weeping...";
    }   };
int main(void) {
```
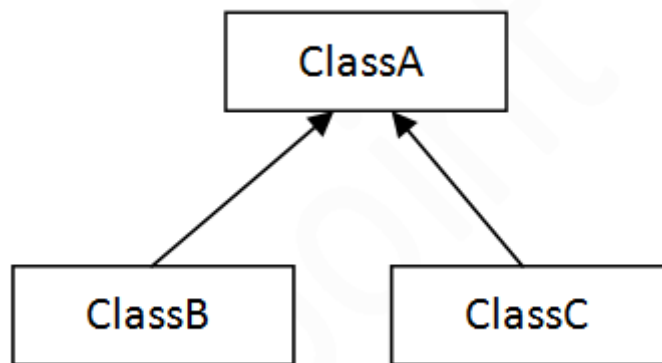
```
      BabyDog d1;
      d1.eat();
      d1.bark();
       d1.weep();
       return 0;
    }
```

   Output:
Eating...
Barking...
Weeping...

Hierarchical Inheritance:

In this type of inheritance, multiple derived classes inherits from a single base class.

Hierarchical Inheritance

```
   #include <iostream>
   using namespace std;
   class Shape  // Declaration of base class.
   {
      public:
      int a;
      int b;
      void get_data(int n,int m)
      {
         a= n;
          b = m;
```

```cpp
} };
class Rectangle : public Shape  // inheriting Shape class
{
    public:
    int rect_area()
    {
        int result = a*b;
        return result;
    } };
class Triangle : public Shape    // inheriting Shape class
{
    public:
    int triangle_area()
    {
        float result = 0.5*a*b;
        return result;
    }
};
int main()
{ Rectangle r;
    Triangle t;
    int length,breadth,base,height;
    cout << "Enter the length and breadth of a rectangle: " << endl;
    cin>>length>>breadth;
    r.get_data(length,breadth);
    int m = r.rect_area();
    cout << "Area of the rectangle is : " <<m<<endl;
    cout << "Enter the base and height of the triangle: " << endl;
    cin>>base>>height;
    t.get_data(base,height);
    float n = t.triangle_area();
    cout <<"Area of the triangle is : " << n<<endl;
```

```
      return 0;
}
```

Output: Enter the length and breadth of a rectangle:
   23
   20
   Area of the rectangle is : 460
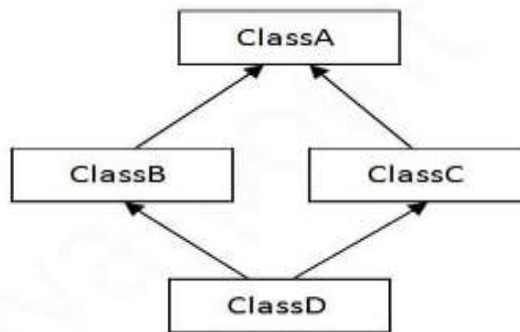   Enter the base and height of the triangle:
   2
   5
   Area of the triangle is : 5

Hybrid Inheritance:

Hybrid Inheritance is combination of any 2 or more types of inheritances

Hybrid Inheritance

```cpp
#include <iostream>
using namespace std;
class A
{  protected:
   int a;
   public:
   void get_a()
   {
     cout << "Enter the value of 'a' : " << endl;
     cin>>a;
   }
};
class B : public A
```

```
   {
      protected:
      int b;
      public:
      void get_b()
      { cout << "Enter the value of 'b' : " << endl;
        cin>>b;
      }
   };
   class C
   { protected:
      int c;
      public:
      void get_c()
      { cout << "Enter the value of c is : " <<endl;
        cin>>c;
      }
   };
   class D : public B, public C
   { protected:
      int d;
      public:
      void mul()
      {
         get_a();
         get_b();
         get_c();
     cout << "Multiplication of a,b,c is : " <<a*b*c<< endl;
      }
   };
   int main()
   { D d;
```

```
        d.mul();
        return 0;
    }
```

Output:
Enter the value of 'a' : 10
Enter the value of 'b' : 20
Enter the value of c is :30
Multiplication of a,b,c is : 6000

Exception Handling: Exception handling is part of C++ and object oriented programming. they are added in C++ to handle the unwanted situations during program execution. If we do not type the program correctly then ot might result in errors. Main purpose of exception handling is to identify and report the runtime error in the program. Famous examples are divide by zero, array index out of bound error, file not found, device not found, etc. C++ exception handling is possible with three keywords iz. try, catch and throw. Exception handling performs the following tasks:-
Find the problem in the given code. It is also called as hit exception.

- It informs error has occurred. It is called as throwing the exception.
- We receive the roe info. It is called as catching the exception.
- It takes the corrective action. It is called as exception handling.

TRY:- It is block code in which there are chances of runtime error. This block is followed by one or more catch block. Most error prone code is added in try block.

CATCH:- This is used to catch th exception thrown by the try block. In catch block we take corrective action on throwing exception. If files are open end , we can take corrective action like closing file handles ,closing database connections ,saving unsaved work ,etc.

THROW:- Program throws exception when problem occurs. It is possible with throw keyword.

SNYTAX:=
```
//normal program code
try{
throw exception
}
catch(argument)
{
...
```

```
...
}
//rest of the code
// Exception Handling
#include <iostream>
using namespace std;
int main()
{
int x = -1;
// Some code
cout<< "Before try \n";
try {
cout<< "Inside try \n";
if (x < 0)
{
throw x;
cout<< "After throw (Never executed) \n";
}
}
catch (int x ) {
cout<< "Exception Caught \n";
}
cout<< "After catch (Will be executed) \n";
return 0;
}
```

Output:
Before try
Inside try
Exception Caught
After catch (Will be executed)

Facilities:
Linux Operating Systems, G++

Algorithm:
1. Start.
2. Create classes Publication, book and tape.
3 .Publication class having data members title, price.
4.Class Book having data members pages and member functions getdata() and pudata().

5.Class Tape having data members minutes and member functions getdata() and pudata().

6. Create an object bof class book and object t of class tape.

7. Stop.

Input: A class publication that stores the title (a string) and price (type float) of publications. Derives two classes Book and Tape.

Output:

Display title and price from publication class. The result in following format:

Enter Title: OOP

Enter Price: 300

Enter Pages: 250

Enter Title: POP

Enter Price: 200

Enter Minutes: 60

Title: OOP

Price: 300

Pages: 250

Title: POP

Price: 200

Minutes: 60

Conclusion:

Hence, we have successfully studied concept of inheritance and exception handling.

Questions:

1. What is Inheritance?

2. What are types of Inheritance?

3. What is Single Inheritance?

4. What is Multiple Inheritance?

5. What is Hierarchical Inheritance?

6. What is Multilevel Inheritance?

7. What is Hybrid Inheritance?

8. What is Exception handling?

9. What are try catch block of exception handling?

GROUP B

Assignment No.	4 (GROUP B)
Title	Write a C++ program that creates an output file, writes information to it, closes the file and open it again as an input file and read the information from the file.
Subject	Object Oriented Programming
Class	S.E. (C.E.)
Roll No.	
Date	
Signature	

Assignment No: 4

Title: File handing

Problem Statement: Write a C++ program that creates an output file, writes information to it, closes the file and open it again as an input file and read the information from the file.

Prerequisites: Object Oriented Programming

Objectives: To learn the concept of file handling.

Theory:

Stream:

A stream is a sequence of bytes. It acts as source from which the input data can be obtained or as a destination to which the output data can be sent.

1. InputStream

Input Streams are used to hold input from a data producer, such as a keyboard, a file, or a network. The source stream that provides data to the program is called the input stream. A program extracts the bytes from the input stream. In most cases the standard input device is the keyboard. With the cin and "extraction" operator (>>) it is possible to read input from the keyboard.

2. OutputStream

Output Streams are used to hold output for a particular data consumer, such as a monitor, a file, or a printer. The destination stream that receives data from the program is called the output stream. A program inserts the bytes into an output stream. By default, the standard output of a program points at the screen. So with the cout operator and the "insertion" operator (<<) you can print a message onto the screen.iostream standard library provides cin and cout methods for reading from standard input and writing to standard output respectively.file handling provides three new datatypes:

Data Type	Description
ofstream	This data type represents the output file stream and is used to create files and to write information to files.
ifstream	This data type represents the input file stream and is used to read information from files.

fstream	This data type represents the file stream generally, and has the capabilities of both ofstream and ifstream which means it can create files, write information to files, and read information from files.

To perform file processing in C++, header file

<iostream> and <fstream> must be included in your C++ source file.

Opening a File

- A file must be opened before you can read from it or write toit.

- Either the ofstream or fstream object may be used to open a file for writing and ifstream object is used to open a file for reading purpose only.

- Following is the standard syntax for open() function which is a member of fstream, ifstream and of stream objects.

  void open(const char *filename, ios::openmode mode);

Here, the first argument specifies the name and location of the file to be opened and the second argument of the open() member function defines the mode in which the file should be opened.

Mode Flag	Description
ios::app	Append mode. In this All output to that file to be appended to theend.
ios::ate	Open a file for output and move the read/write control to the end of the file.
ios::in	Open a file for reading.
ios::out	Open a file for writing.
ios::trunk	If the file already exists, its contents will be truncated before opening the file.

- You can combine two or more of these values by ORing them together.

- For example, if you want to open a file in write mode and want to truncate it incase it already exists, following will be the syntax:

```
ofstream outfile;
fstreamafile;
afile.open("file.dat", ios::out | ios::in );
```

- Similar way, you can open a file for reading and writing purpose as follows:

Closing a File:

When a C++ program terminates it automatically closes flushes all the streams,release all the allocated memory and close all the opened files. It is always a good practice that a programmer should close all the opened files before program termination. following is the standard syntax for close( ) function, which is a member of fstream ,ifstream, and ofstream objects.

```
void close( );
```

Writing to a File
• While doing C++ programming, you write information to a file from your program using the stream insertion operator (<<) just as you use that operator to output information to the screen.
• The only difference is that you use an ofstream or fstream object instead of the cout object.

Reading from a File
• You read information from a file into your program using the stream extraction operator (>>) just as you use that operator to input information from the keyboard.
• The only difference is that you use an ifstream or fstream object instead of the cin object.

```
file .read ((char *)&V , sizeof (V)); file . Write ((char *)&V , sizeof (V));
```

Facilities:
Linux Operating Systems, G++
Algorithm:
1. Start
2. Create a class
3. Define data members roll number and name.
4. Define accept() to take name and roll number from user.
5. Define display() to display the record.
6. In main() create the object of class and fstream class.

7.Take a limit from user in n variable.

8. Open the file in out mode , call accept() to take record from user, then call write() to write that record into the file and at the end close that file.

9. Open the file in in mode, read the record from the file ,call display() function to display the record and at the end close that file.

10. Stop

Input:
how many record you want 3
1 abc
2 pqr
3 xyz

Output:
name=abc
Roll=1
name=pqr
Roll=2
name=xyz
Roll=3

Conclusion:
Hence, we have studied concept of File handing

Questions:
1. What is file handling?
2. What are the different benefits of file handling?
3. What is fstream class?
4. How to create object of fsream class?
5. Explain the syntax of read() ?
6. Explain the syntax of write()?

Assignment No.	5 (GROUP B)
Title	Write a function template selection sort. Write a program that inputs, sorts and outputs an integer array and a float array.
Subject	Object Oriented Programming
Class	S.E. (C.E.)
Roll No.	
Date	
Signature	

Assignment No: 5

Title: Function Template

Problem Statement: Implement a function template selection Sort. Write a program that inputs, sorts and outputs an integer array and a float array.

Prerequisites: Object Oriented Programming

Objectives: To learn the concept of file Template.

Theory:
Templates
Templates are the foundation of generic programming, which involves writing code in a way that is independent of any particular type.

A template is a blueprint or formula for creating a generic class or a function. The library containers like iterators and algorithms are examples of generic programming and have been developed using template concept. There is a single definition of each container, such as vector, but we can define many different kinds of vectors for example, vector <int> or vector <string>.

You can use templates to define functions as well as classes, let us see how do they work:
Function Template:
The general form of a template function definition is shown here:
template <class type> ret-type func-name(parameter list)
{
// body of function
}
Here, type is a placeholder name for a data type used by the function. This name can be used within the function definition.
The following is the example of a function template that returns the maximum of two values:
#include<iostream>
#includede<string>
using namespace std;
<typename T>
inline T const& Max (T const& a, T const& b)
{
return a < b ? b:a;
}

```
int main ()
{
inti = 39; int j = 20;
cout<< "Max(i, j): " << Max(i, j) <<endl; double f1 =13.5;
double f2 =20.7;
cout<< "Max(f1, f2): " << Max(f1, f2) <<endl; string s1 = "Hello";
string s2 = "World";
cout<< "Max(s1, s2): " << Max(s1, s2) <<endl; return 0;
}
```

If we compile and run above code, this would produce the following result:

Max(i, j): 39
Max(f1, f2): 20.7
Max(s1, s2): World

Class Template:

Just as we can define function b templates, we can also define class templates. The general form of a generic class declaration is shown here:

```
template <class type> class class-name
{
.
.
.
}
```

Here, type is the placeholder type name, which will be specified when a class is instantiated. You can define more than one generic data type by using a comma-separated list. Following is the example to define class Stack<> and implement generic methods to push and pop
the elements from the stack:

```
#include <iostream>
#include <vector>
#include <cstdlib>
#include <string>
#include <stdexcept>
using namespace std;
template <class T> class Stack
{
private:
vector<T>elems; // elements
```

```
public:
void push(T const&); //push element
voidpop(); // pop element
Ttop()const; // return top element bool empty()const
{ // return true if empty. returnelems.empty();
}
};
template <class T>
void Stack<T>::push (T const&elem)
{
// append copy of passed element
elems.push_back(elem);
}
template<class T>
void Stack<T>::pop ()
{
if (elems.empty())
{
throw out_of_range("Stack<>::pop(): empty stack");
}
// remove last element
elems.pop_back();
}
template <class T>
T Stack<T>::top () const
{
if (elems.empty())
{
throw out_of_range("Stack<>::top(): empty stack");
}
// return copy of last element
return elems.back();
}
int main()
{
try
{
```

```
Stack<int> intStack; // stack of ints
Stack<string>stringStack; // stack ofstrings
// manipulate int stack
int Stack.push(7);
cout<<intStack.top() <<endl;
// manipulate string stack
stringStack.push("hello");
cout<<stringStack.top() <<std::endl; stringStack.pop();
stringStack.pop();
}
catch (exception const& ex)
{
cerr<< "Exception: " <<ex.what() <<endl; return -1;
}
}
```

If we compile and run above code, this would produce the following result:

7

hello

Exception: Stack<>::pop(): empty stack

Selection Sort:

Selection sort is a sorting algorithm, specifically an in-place comparison sort. It has O(n2) time complexity, making it inefficient on large lists, and generally performs worse than the similar insertion sort. Selection sort is noted for its simplicity, and it has performance advantages over more complicated algorithms in certain situations, particularly where auxiliary memory is limited

How selection sort works?

Example

For the first position in the sorted list, the whole list is scanned sequentially. The first position where 14 is stored presently, we search the whole list and find that 10 is the lowest value.

So we replace 14 with 10. After one iteration 10, which happens to be the minimum value in the list, appears in the first position of sorted list.

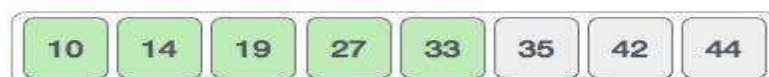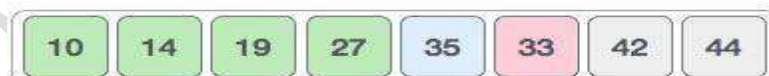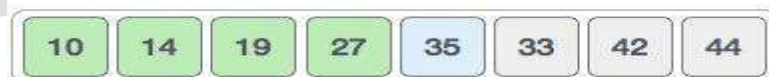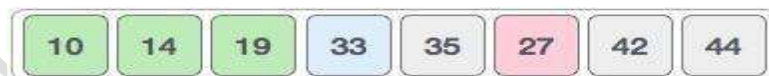| 10 | 33 | 27 | 14 | 35 | 19 | 42 | 44 |

For the second position, where 33 is residing, we start scanning the rest of the list in linear manner.

| 10 | 33 | 27 | 14 | 35 | 19 | 42 | 44 |

We find that 14 is the second lowest value in the list and it should appear at the second place. We swap these values

| 10 | 33 | 27 | 14 | 35 | 19 | 42 | 44 |

After two iterations, two least values are positioned at the beginning in the sorted manner.
The same process is applied on the rest of the items in the array. Pictorial depiction of entire sorting process is as follows –

| 10 | 14 | 27 | 33 | 35 | 19 | 42 | 44 |

| 10 | 14 | 27 | 33 | 35 | 19 | 42 | 44 |

| 10 | 14 | 19 | 33 | 35 | 27 | 42 | 44 |

| 10 | 14 | 19 | 33 | 35 | 27 | 42 | 44 |

| 10 | 14 | 19 | 27 | 35 | 33 | 42 | 44 |

| 10 | 14 | 19 | 27 | 35 | 33 | 42 | 44 |

| 10 | 14 | 19 | 27 | 35 | 33 | 42 | 44 |

| 10 | 14 | 19 | 27 | 33 | 35 | 42 | 44 |

| 10 | 14 | 19 | 27 | 33 | 35 | 42 | 44 |

Facilities:
Linux Operating Systems, G++
Algorithm:
1. Start
2. Declare the template parameterT.
3. Define template function for selection sort.
4. In main() Define two arrays, one for integer and another for float. and take a input for both the arrays and call sorting function template to sort the number.
5. Stop
Input:
Selection sort Integer Element
Enter how many elements you want 5
Enter the Integer element 7
5
8
9
3
Float Element
Enter how many elements you want 5
Enter the float element 3.8
9.4
5.5
2.2
6.7
Output:
Sorted list=
3 5 7 8 9
Sorted list=
2.2 3.8 5.5 6.7 9.4
Conclusion:
Hence, we have studied concept of Function Template.
Questions:
1. What is template?
2. What is Function template?
3. What is Class template?
4. Explain template with function overloading.
5. Explain template with non-type argument.

GROUP C

Assignment No.	6 (GROUP C)
Title	Write C++ program using STL for sorting and searching user defined records such as personal records (Name, DOB, Telephone number etc) using vector container. **OR** Write C++ program using STL for sorting and searching user defined records such as Item records (Item code, name, cost, quantity etc) using vector container.
Subject	Object Oriented Programming
Class	S.E. (C.E.)
Roll No.	
Date	
Signature	

Assignment No:6

Title: Personnel information system using sorting and searching for STL and vector container.

Problem Statement: Write C++ program using STL for sorting and searching user defined records such as personal records (Name, DOB, Telephone number etc) using vector container.

OR

Write C++ program using STL for sorting and searching user defined records such as Item records (Item code, name, cost, quantity etc) using vector container.

Prerequisites: Object Oriented Programming

Objectives: To learn the concept STL, searching, sorting and vector container.

Theory:

STL:

The Standard Template Library (STL) is a set of C++ template classes to provide common programming data structures and functions such as lists, stacks, arrays, etc. It is a library of container classes, algorithms, and iterators. It is a generalized library and so, its components are parameterized. A working knowledge of template classes is a prerequisite for working with STL.

STL has four components
• Algorithms
• Containers
• Functions
• Iterators

Algorithms

• The algorithm defines a collection of functions especially designed to be used on ranges of elements. They act on containers and provide means for various operations for the contents of the containers.
• Algorithm
• Sorting
• Searching

- Important STL Algorithms
- Useful Array algorithms
- Partition Operations
- Numeric

Containers
•Containers or container classes store objects and data. There are in total seven standard "first-class" container classes and three container adaptor classes and only seven header files that provide access to these containers or container adaptors.
- Sequence Containers: implement data structures which can be accessed in a sequential manner.
- vector
- list
- deque
- arrays
- forward_list(Introduced in C++11)

- **Container Adaptors :** provide a different interface for sequential containers. • queue
- priority_queue
- stack

- **Associative Containers :** implement sorted data structures that can be quickly searched (O(log n) complexity).
- set
- multiset
- map
- multimap

- **Unordered Associative Containers :** implement unordered data structures that can be quickly searched
- unordered_set
- unordered_multiset
- unordered_map
- unordered_multimap

Functions
- The STL includes classes that overload the function call operator. Instances of such classes are called function objects or functors. Functors allow the working of the associated function to be customized with the help of parameters to be passed.

Iterators
• As the name suggests, iterators are used for working upon a sequence of values. They are the major feature that allow generality in STL.

Utility Library
• Defined in header <utility>.
• pair

Sorting:
It is one of the most basic functions applied to data. It means arranging the data in a particular fashion, which can be increasing or decreasing. There is a built in function in C++ STL by the name of sort( ). This function internally uses Intro Sort. In more details it is implemented using hybrid of Quick Sort,
Heap Sort and Insertion Sort. By default, it uses Quick Sort but if Quick Sort is doing unfair partitioning and taking more than N*log N time, it switches to Heap Sort and when the array size becomes really small, it switches to Insertion Sort. The prototype for sort is :

sort(startaddress, endaddress)

startaddress: the address of the first element of the array

endaddress: the address of the next contiguous location of the last element of the array.

So actually sort() sorts in the range of [startaddress,endaddress)

```
//Sorting

  #include <iostream>
  #include <algorithm>
  using namespace std;
void show(int a[])
{
  for(inti = 0; i < 10; ++i)
    cout<< a[i] << " ";
}
int main()
{
  int a[10]= {1, 5, 8, 9, 6, 7, 3, 4, 2, 0};
```

```
    cout<< "\n The array before sorting is :";
    show(a);
    sort(a, a+10);
    cout<< "\n\n The array after sorting is : ";
    show(a);
    return0;
 }
```

Output:
The output of the above program is :
The array before sorting is : 1 5 8 9 6 7 3 4 2 0
The array after sorting is : 0 1 2 3 4 5 6 7 8 9

Searching:
It is a widely used searching algorithm that requires the array to be sorted before search is applied. The main idea behind this algorithm is to keep dividing the array in half (divide and conquer) until the element is found, or all the elements are exhausted.
It works by comparing the middle item of the array with our target, if it matches, it returns true otherwise if the middle term is greater than the target, the search is performed in the left sub-array. If the middle term is less than target, the search is performed in the right sub-array.
The prototype for binary search is :

binary_search(startaddress, endaddress, valuetofind)

startaddress: the address of the first element of the array.

endaddress: the address of the last element of the array.

Value to find: the target value which we have to search for.

//Searching
```
#include <algorithm>
#include <iostream>
using namespace std;
void show(int a[], int arraysize)
{ for(int i = 0; i <arraysize; ++i)
cout<< a[i] << " ";
}
int main() {
int a[] = { 1, 5, 8, 9, 6, 7, 3, 4, 2, 0 };
```

```
int asize = sizeof(a) / sizeof(a[0]);
cout<< "\n The array is : ";
show(a, asize);
cout<< "\n\nLet's say we want to search for 2 in the array";
cout<< "\n So, we first sort the array";
sort(a, a + asize);
cout<< "\n\n The array after sorting is : ";
show(a, asize);
cout<< "\n\nNow, we do the binary search";
if(binary_search(a, a + 10, 2))
cout<< "\nElement found in the array";
else
cout<< "\nElement not found in the array";
cout<< "\n\nNow, say we want to search for 10";
if(binary_search(a, a + 10, 10))
cout<< "\nElement found in the array";
else
cout<< "\nElement not found in the array";
return0;
}
```

Output:

The array is : 1 5 8 9 0 6 7 3 4 2 0

Let's say we want to search for 2 in the array
So, we first sort the array
The array after sorting is : 0 1 2 3 4 5 6 7 8 9
Now, we do the binary search
Element found in the array
Now, say we want to search for 10Element not found in the array

Facilities:

Linux Operating Systems, G++

Algorithm:

1. Start.
2. Give a header file to use 'vector'.
3. Create a vector naming 'personal_records'.
4. Initialize variables to store name, birth date and telephone number.

5. Using iterator store as many records you want to store using predefined functions as push_back().
6. Create another vector 'item_record'
7. Initialize variables to store item code, item name, quantity and cost.
8. Using iterator and predefined functions store the data.
9. Using predefined function sort(), sort the data stored according to user requirements.
10. Using predefined function search, search the element from the vector the user wants to check.
11. Display and call the functions using a menu**.**
12. End.

Input:
Personnel information such as name, DOB, telephone number.

Output:
Display personnel information from database. The result in following format:
***** Menu *****
1.Insert
2.Display
3.Search
4.Sort
5.Delete
6.Exit
Enter your choice:1
Enter Item Name: bat
Enter Item Quantity:2
Enter Item Cost:50
Enter Item Code:1

Conclusion:
Hence, we have successfully studied the concept of STL(Standard Template Library) and how it makes many data structures easy. It briefs about the predefined functions of STL and their uses such a search() and sort()

Questions:
1. What is STL?
2. What are four components of STL?
3. What is Sorting?
4. What is Searching?
5. What vector container?

Assignment No.	7 (GROUP C)
Title	Write a program in C++ to use map associative container. The keys will be the names of states and the values will be the populations of the states. When the program runs, the user is prompted to type the name of a state. The program then looks in the map, using the state name as an index and returns the population of the state.
Subject	Object Oriented Programming
Class	S.E. (C.E.)
Roll No.	
Date	
Signature	

Assignment No: 7

Title: To use map associative container.

Problem Statement: Write a program in C++ to use map associative container. The keys will be the names of states and the values will be the populations of the states. When the program runs, the user is prompted to type the name of a state. The program then looks in the map, using the state name as an index and returns the population of the state

Prerequisites: Object Oriented Programming

Objectives: To learn the concept of map associative container.

Theory:

Map associative container:

Map associative container are associative containers that store elements in a mapped fashion. Each element has a key value and a mapped value. No two mapped values can have same key values.

map::operator[]

This operator is used to reference the element present at position given inside the operator. It is similar to the at() function, the only difference is that the at() function throws an out-of-range exception when the position is not in the bounds of the size of map, while this operator causes undefined behaviour.

mapname[*key*]**Parameters :**

Key value mapped to the element to be fetched.

Returns :

Direct reference to the element at the given key value.

Examples:

Input : map mymap;

mymap['a'] = 1;

mymap['a'];

Output : 1

Input : map mymap;

mymap["abcd"] = 7;

mymap["abcd"];

Output : 7

//Program

```
#include <map>
#include <iostream>
#include<string>
using namespace std;
int main()
{
// map declaration
map<int,string>mymap;
// mapping integers to strings
mymap[1] = "Hi";
mymap[2] = "This";
mymap[3] = "is";
mymap[4] = "SCOE";
// using operator[] to print string
// mapped to integer 4
cout<<mymap[4];
return0;
}
```
Output: SCOE

Facilities:
Linux Operating Systems, G++
Algorithm:
1. Start.
2. Give a header file to map associative container.
3. Insert states name so that we get values as population of that state.
4. Use population Map. insert().
5. Display the population of states**.**
6. End.

Input:

Information such as state name to map associative container.

Output:
 Size of population Map:5

Brasil: 193 illion
China: 1339 million

India: 1187 million

Indonesia: 234 million
Pakistan: 170 million
Indonesia's populations is 234 million

Conclusion:
Hence, we have successfully studied the concept of map associative container
Questions:

1. What is an associative container in C++?
2. What is map in C++?
3. How to do declare a map?
4. Explain Associative mapping with example?

PART-II COMPUTER GRAPHICS

Suggested List of Laboratory Experiments/Assignments
(**All assignments are compulsory**)

GROUP A

Assignment No.	1 (GROUP A)
Title	A concave polygon filling using scan fill algorithm
Subject	Computer Graphics Laboratory
Class	S.E. (C.E.)
Roll No.	
Date	
Signature	

Assignment No. 1

Title: A concave polygon filling using scan fill algorithm

Problem Statement: Write C++ program to draw a concave polygon and fill it with desired color using scan fill algorithm. Apply the concept of inheritance.

Prerequisites: 1. Basic programming skills of C++

Objectives: To understand and implement scanline polygon fill algorithm.

Theory:
  Polygon:

  A polygon is a closed planar path composed of a finite number of sequential line segments. Apolygon is a two-dimensional shape formed with more than three straight lines. When starting point and terminal point is same then it is called polygon.
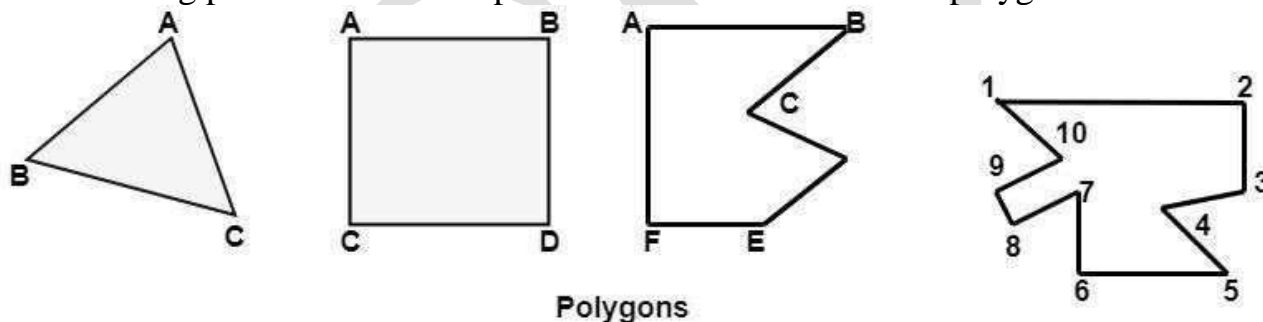
Polygons

Types of Polygons
Concave
Convex
Complex

A convex polygon is a simple polygon whose interior is a convex set. In a convex polygon, all interior angles are less than 180 degrees.

The following properties of a simple polygon are all equivalent to convexity:

• 　　Every internal angle is less than or equal to 180 degrees.

- Every line segment between two vertices remains inside or on the boundary of thepolygon.

Convex Polygons: In a convex polygon, any line segment joining any two inside points lies insidethe polygon. A straight line drawn through a convex polygon crosses at most two sides.

A concave polygon will always have an interior angle greater than 180 degrees. It is possible tocut
a concave polygon into a set of convex polygons. You can draw at least one straight line through a
concave polygon that crosses more than two sides.

Complex polygon is a polygon whose sides cross over each other one or more times.

Convex polygon   Concave polygon   Complex Polygon
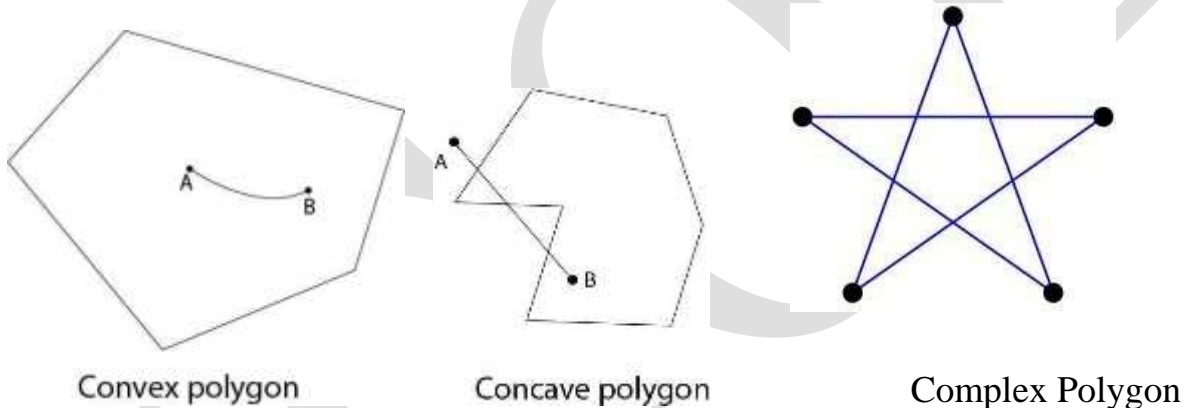
Inside outside test (Even- Odd Test):
We assume that the vertex list for the polygon is already stored and proceed as follows.
1. Draw any point outside the range X_{min} and X_{max} and Y_{min} and Y_{max}. Draw a scan line
through P up toapoint A under study

(Xmax, Ymax)

A — P

(Xmin, Ymin)
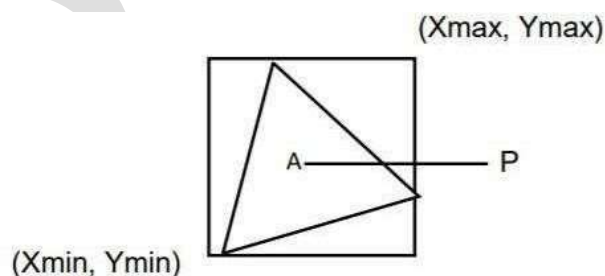
2. If this scan line
i) Does not pass through any of the vertices then its contribution is equal to the number of

times itintersects the edges of the polygon. Say C if

a) C is odd then A lies inside the polygon.

b) C is even then it lies outside the polygon.

ii) If it passes through any of the vertices then the contribution of this intersection say V is,

a) Taken as 2 or even. If the other points of the two edges lie on one side of the scan line.

b) Taken as 1 if the other end points of the 2 edges lie on the opposite sides of the scan-line. c)Here will be total contribution is C + V.

Polygon Filling:

For filling polygons with particular colors, you need to determine the pixels falling on the border of the polygon and those which fall inside the polygon.

Scan fill algorithm:

A scan-line fill of a region is performed by first determining the intersection positions of the boundaries of the fill region with the screen scan lines n Then the fill colors are applied to each sectionof a scan line that lies within the interior of the fill region n The scan-line fill algorithm identifies thesame interior regions as the odd-even rule.

It is an image space algorithm. It processes one line at a time rather than one pixel at a time. It uses the concept area of coherence. This algorithm records edge list, active edge list. So accuratebookkeeping is necessary. The edge list or edge table contains the coordinate of two endpoints. ActiveEdge List (AEL) contain edges a given scan line intersects during its sweep. The active edge list (AEL) should be sorted in increasing order of x. The AEL is dynamic, growing and shrinking
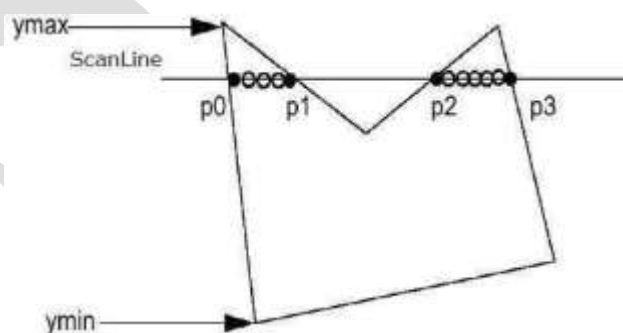
Algorithm

Step1: Start algorithm

Step2: Initialize the desired data structure

1. Create a polygon table having color, edge pointers, coefficients
2. Establish edge table contains information regarding, the endpoint of edges, pointer to polygon, inverse slope.
3. Create Active edge list. This will be sorted in increasing order of x.
4. Create a flag F. It will have two values either on or off.

Step3: Perform the following steps for all scan lines

1. Enter values in Active edge list (AEL) in sorted order using y as value
2. Scan until the flag, i.e. F is on using a background color
3. When one polygon flag is on, and this is for surface S1enter color intensity as I1into refreshbuffer
4. When two or image surface flag are on, sort the surfaces according to depth and useintensity value Sn for the nth surface. This surface will have least z depth value
5. Use the concept of coherence for remaining planes.

Step4: Stop Algorithm

Input: Enter the no. of edges

100 200 300 400 500 50

Output:

Conclusion: Hence, we have studied concept of concave scan line algorithm.

Questions:
1. Which are the different approaches to fill a polygon?
2. What are advantages and drawbacks of scan line polygon fill algorithm?

Assignment No.	2 (GROUP A)
Title	Polygon clipping using Cohen Southerland line clipping algorithm
Subject	Computer Graphics Laboratory
Class	S.E. (C.E.)
Roll No.	
Date	
Signature	

Assignment No. 2

Title: Polygon clipping using Cohen Southerland line clipping algorithm

Problem Statement: Write C++ program to implement Cohen Southerland line clipping algorithm.

Prerequisites: 1. Basic programming skills of C++

Objectives: To learn Cohen Southerland line clipping algorithm.

Theory:
   Cohen Sutherland Algorithm is a **line clipping algorithm** that cuts lines to portions which are within a rectangular area. It eliminates the lines from a given set of lines and rectangle area of interest(view port) which belongs outside the area of interest and clips those lines which are partially inside the area of interest.
   Example:

Cohen Sutherland Line Clipping Algorithm
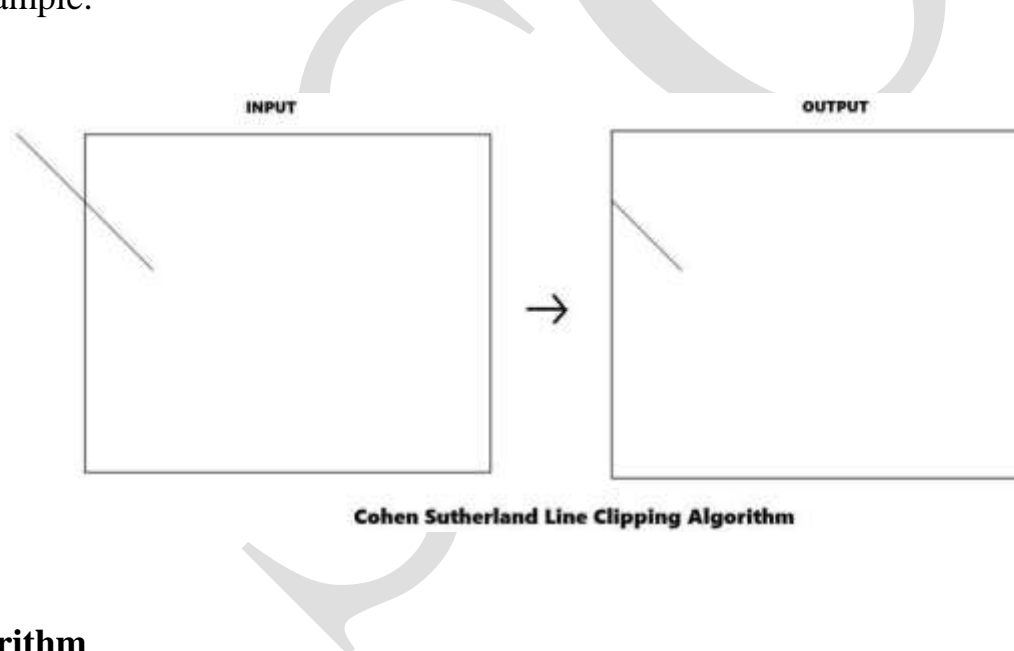
Algorithm

   The algorithm divides **a two-dimensional space** into **9 regions** (eight outside regions and one inside region) and then efficiently determines the lines and portions of lines that are visible in the central region of interest (the viewport).

Following image illustrates the 9 regions:
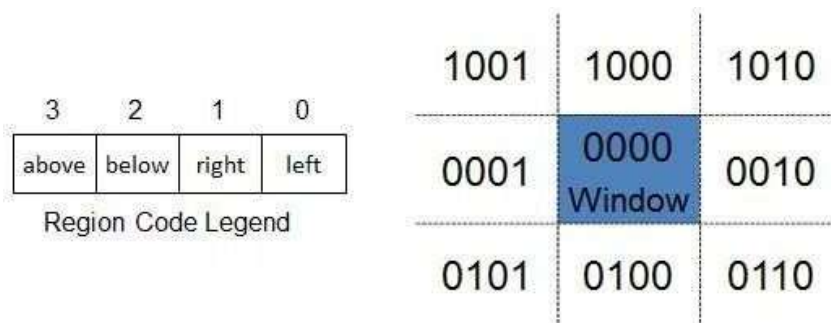
As you seen each region is denoted by a 4 bit code like 0101 for the bottom right region

Four Bit code is calculated by comparing extreme end point of given line (x,y) by four co-ordinates x_min, x_max, y_max, y_min which are the coordinates of the area of interest (0000) Calculate the four bit code as follows:

- Set First Bit if 1 Points lies to left of window (**x < x_min**)
- Set Second Bit if 1 Points lies to right of window (**x > x_max**)
- Set Third Bit if 1 Points lies to left of window (**y < y_min**)
- Set Forth Bit if 1 Points lies to right of window (**y > y_max**)
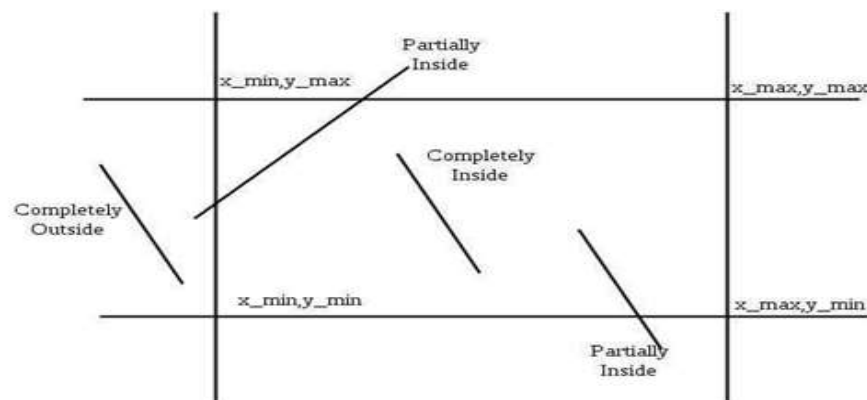
The more efficient Cohen-Sutherland Algorithm performs initial tests on a line to determine whether intersection calculations can be avoided.

ALgorithm

- **Step 1** : Assign a region code for two endpoints of given line
- **Step 2** : If both endpoints have a region code 0000 then given line is completely inside andwe will keep this line.
- **Step 3:** If step 2 fails, perform the logical AND operation for both region codes.
- **Step 3.1:** If the result is not 0000, then given line is completely outside.
- **Step 3.2 :** Else line is partially inside.
- **Step 3.2.a :** Choose an endpoint of the line that is outside the given rectangle.
- **Step 3.2.b :** Find the intersection point of the rectangular boundary(based on region code)
- **Step 3.2.c :** Replace endpoint with the intersection point and upgrade the region code.
- **Step 3.2.d :** Repeat step 2 until we find a clipped line either trivially accepted or rejected.
- **Step 4:** Repeat step 1 for all lines.

Input: Enter the coordinates : 200 100 300
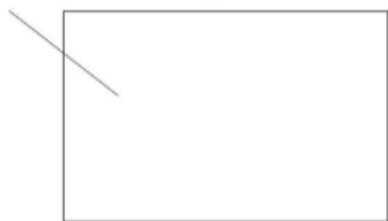
Output:
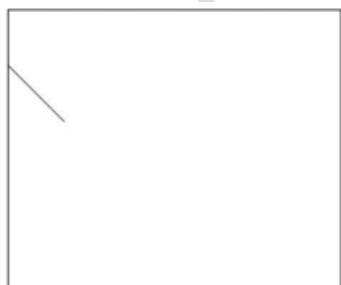
Conclusion: Hence, we have studied concept of polygon clipping using Cohen Southerland line clipping algorithm.

Questions:

1. What is the limitation of Cohen Sutherland Line Clipping algorithm?
2. What are the advantages of Cohen Sutherland Line Clipping?

Assignment No.	3 (GROUP A)
Title	Pattern drawing using line and circle.
Subject	Computer Graphics Laboratory
Class	S.E. (C.E.)
Roll No.	
Date	
Signature	

Assignment No. 3

Title: Pattern drawing using line and circle.

Problem Statement: Write C++ program to draw a given pattern. Use DDA line and Bresenham's circle drawing algorithm. Apply the concept of encapsulation.

Prerequisites: 1. Basic programming skills of C++

Objectives: To learn and apply DDA line and Bresenham's circle drawing algorithm

Theory:

DDA Line Drawing Algorithm:

Pixel is a basic element in graphics. To draw a line, you need two end points between which you can draw a line. Digital Differential Analyzer (DDA) line drawing algorithm is the simplest line drawing algorithm in computer graphics. It works on incremental method. It plots the points from starting point of line to end point of line by incrementing in X and Y direction in each iteration.

DDA line drawing algorithm works as follows:

Step 1: Get coordinates of both the end points (X1, Y1) and (X2, Y2) from user.

Step 2: Calculate the difference between two end points in X and Y

    direction.dx = X2 – X1;
    dy = Y2 – Y1;

Step 3: Based on the calculated difference in step-2, you need to identify the number of steps to putpixel. If dx > dy, then you need more steps in x coordinate; otherwise in y coordinate.

    if (absolute(dx) > absolute(dy))
    Steps = absolute(dx);
    else
    Steps = absolute(dy);

Step 4: Calculate the increment in x coordinate and y coordinate.

Xincrement = dx / (float) steps;
Yincrement = dy / (float) steps;

Step 5: Plot the pixels by successfully incrementing x and y coordinates accordingly and completethe drawing of the line.

```
for(int i=0; i < Steps; i++)
{
        X1 = X1 + Xincrement;
        Y1 = Y1 + Yincrement;
        putpixel(Round(X1), Round(Y1), ColorName);
}
```

Bresenham's Circle Drawing Algorithm:

Circle is an eight-way symmetric figure. The shape of circle is the same in all quadrants. In each quadrant, there are two octants. If the calculation of the point of one octant is done, then the other seven points can be calculated easily by using the concept of eight-way symmetry. Bresenham's Circle Drawing Algorithm is a circle drawing algorithm that selects the nearest pixel position to complete the arc. The unique part of this algorithm is that is uses only integer arithmetic which makes it significantly faster than other algorithms using floating point arithmetic.
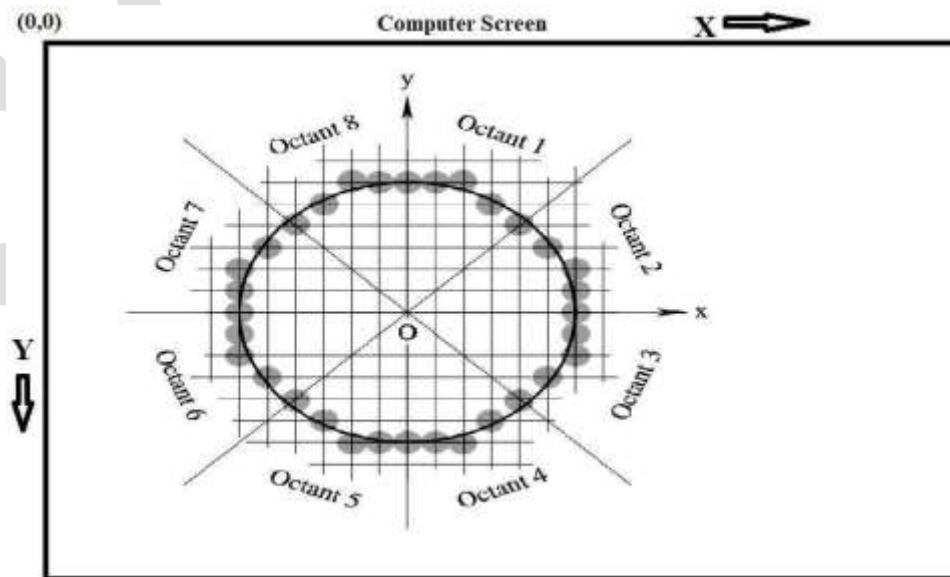
Step 1: Read the x and y coordinates of center: (centx, centy)

Step 2: Read the radius of circle: (r)

Step 3: Initialize, x = 0; y = r;

Step 4: Initialize decision parameter: p = 3 – (2 * r)
Step 5:
```
      do {
putpixel(centx+x, centy-y, ColorName);
if (p<0)
p = p+(4*x)+6;else
{
p=p+[4*(x-y)]+10;
y=y-1;
}
x=x+1;
} while(x<y)
```
Here in step 5, putpixel() function is used which will print Octant-1 of the circle with radius r after the completion of all the iterations of do-while loop. Because of the eight-way symmetry property of circle, we can draw other octants of the circle using following putpixel() functions:

Octant 1: putpixel(centx+x, centy-y, ColorName);

Octant 2: putpixel(centx+y, centy-x, ColorName);

Octant 3: putpixel(centx+y, centy+x, ColorName);

Octant 4: putpixel(centx+x, centy+y, ColorName);
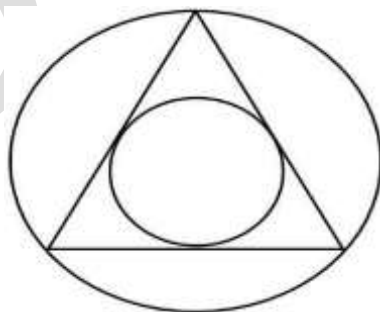
Octant 5: putpixel(centx-x, centy+y, ColorName);

Octant 6: putpixel(centx-y, centy+x, ColorName);

Octant 7: putpixel(centx-y, centy-x, ColorName);

**Octant 8:** putpixel(centx-x, centy-y, ColorName);

Drawing Pattern using lines and circle s:

This pattern is made up of one equilateral triangle and two concentric circles. To draw the triangle, we require coordinates of 3 vertices forming an equilateral triangle. To draw two concentric circles, we require coordinates of common center and radius of both the circles.

We will take coordinates of circle and radius of one of the circle from user. Then using the properties of an equilateral triangle, we can find all 3 vertices of equilateral triangle and radius of other circle. Once we get all these parameters, we can call DDA line drawing and Bresenham's circledrawing algorithms by passing appropriate parameters to get required pattern.

OR
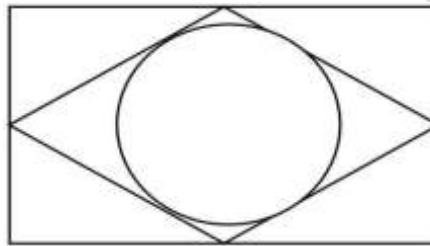
To draw this pattern, we require two rectangles and one circle. We can use suitable geometry to get coordinates to draw rectangles and circle. Then we can call DDA line drawing and Bresenham's circle drawing algorithms by passing appropriate parameters to get required pattern.
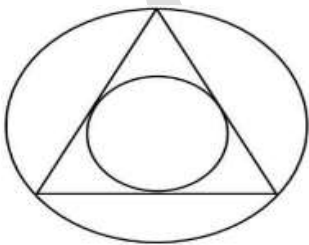
Input: Enter the co-ordinates of center of circle

Value Of X: 100
Value Of Y: 70
Value Of Radius: 30

Output:

Conclusions: Hence, we have studied that DDA line and Bresenham'scircle drawing algorithm and apply the concept of encapsulation.

Questions:

1. Explain the derivation of decision parameters in Bresenham's circle drawing algorithm.
2. Explain the concept of encapsulation with example.

GROUP B

Assignment No.	4 (GROUP B)
Title	Basic 2-D Transformations.
Subject	Computer Graphics Laboratory
Class	S.E. (C.E.)
Roll No.	
Date	
Signature	

Assignment No. 4

Title: Basic 2-D Transformations.

Problem Statement: a) Write C++ program to draw 2-D object and perform following basic transformations: Scaling, Translation, Rotation. Apply the concept of operator overloading.

OR

b) Write C++ program to implement translation, rotation and scaling transformations on equilateral triangle and rhombus. Apply the concept of operator overloading.

.

Prerequisites: Basic programming skills of C++

64-bit Open source Linux

Open Source C++ Programming tool like G++/GCC

**Objectives:** To learn and apply basic transformations on 2-D objects.

Theory:

Transformation means changing some graphics into something else by applying rules. We can have various types of transformations such as translation, scaling up or down, rotation, shearing, reflection etc. When a transformation takes place on a 2D plane, it is called 2D transformation. Transformations play an important role in computer graphics to reposition the graphics on the screen and change their size or orientation. Translation, Scaling and Rotation are basic transformations.

1)   Translation:

A translation moves an object to a different position on the screen. You can translate a point in 2D by adding translation coordinate or translation vector (T_x, T_y) to the original coordinates. Consider

- Initial coordinates of the object $O = (X_{old}, Y_{old})$
- New coordinates of the object O after translation $= (X_{new}, Y_{new})$
- Translation vector or Shift vector $= (T_x, T_y)$
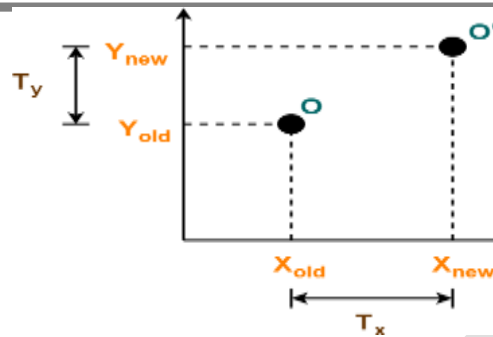
This translation is achieved by adding the translation coordinates to the old coordinates of the object as-

$Xnew = Xold + Tx$ (This denotes translation towards X axis)

$Ynew = Yold + Ty$ (This denotes translation owards Y axis)

In Matrix form, the above translation equations may be represented as-

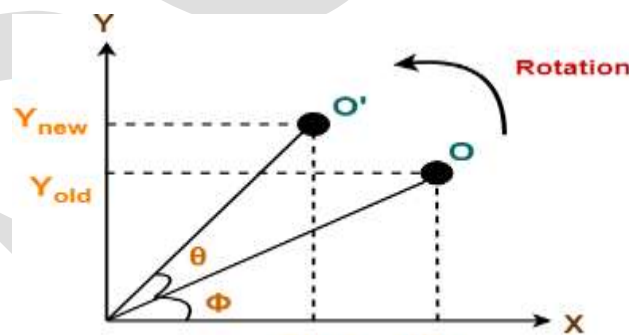$$\begin{bmatrix} X_{new} \\ Y_{new} \end{bmatrix} = \begin{bmatrix} X_{old} \\ Y_{old} \end{bmatrix} + \begin{bmatrix} T_x \\ T_y \end{bmatrix}$$

Translation Matrix

2) Rotation:

In rotation, we rotate the object at particular angle θ (theta) from its original position. Consider
-       Initial coordinates of the object $O = (Xold, Yold)$
-       Initial angle of the object O with respect to origin $= \Phi$
-       Rotation angle $= \theta$
-       New coordinates of the object O after rotation $= (Xnew, Ynew)$

This anti-clockwise rotation is achieved by using the following rotation

equations-$Xnew = Xold \times cos\theta - Yold \times sin\theta$

$Ynew = Xold \times sin\theta + Yold \times cos\theta$
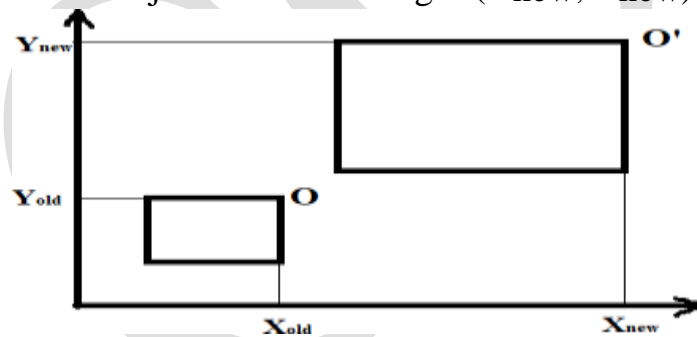
In Matrix form, the above rotation equations may be represented as-

$$\begin{bmatrix} X_{new} \\ Y_{new} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \times \begin{bmatrix} X_{old} \\ Y_{old} \end{bmatrix}$$

Rotation Matrix

3) Scaling:

Scaling transformation is used to change the size of an object. In the scaling process, you either expand or compress the dimensions of the object. Scaling can be achieved by multiplying the original coordinates of the object with the scaling factor (S_x, S_y). If scaling factor > 1, then the object size is increased. If scaling factor < 1, then the object size is reduced. Consider Initial coordinates of the object $O = (X_{old}, Y_{old})$

-      Scaling factor for X-axis = S_x
-      Scaling factor for Y-axis = S_y
-      New coordinates of the object O after scaling = (X_{new}, Y_{new})

This scaling is achieved by using the following scaling equations-

$X_{new} = X_{old} \times S_x$

$Y_{new} = Y_{old} \times S_y$

In Matrix form, the above scaling equations may be represented as-

$$\begin{bmatrix} X_{new} \\ Y_{new} \end{bmatrix} = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \times \begin{bmatrix} X_{old} \\ Y_{old} \end{bmatrix}$$

Scaling Matrix

Homogeneous Coordinates:

Matrix multiplication is easier to implement in hardware and software as compared to matrix addition. Hence we want to replace matrix addition by multiplication while performing transformation operations. So the solution is **homogeneous coordinates,** which allows us to express all transformations (including translation) as matrix multiplications.

To obtain homogeneous coordinates we have to represent transformation matrices in 3x3 matrices instead of 2x2. For this we add dummy coordinate. Each 2 dimensional position (x,y) can be represented by homogeneous coordinate as (x,y,1).

Translation Matrix (Homogeneous Coordinates representation)

$$
\begin{bmatrix} X_{new} \\ Y_{new} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & T_x \\ 0 & 1 & T_y \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} X_{old} \\ Y_{old} \\ 1 \end{bmatrix}
$$

Rotation Matrix (Homogeneous Coordinates representation)

$$
\begin{bmatrix} X_{new} \\ Y_{new} \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} X_{old} \\ Y_{old} \\ 1 \end{bmatrix}
$$

Scaling Matrix (Homogeneous Coordinates representation)

$$
\begin{bmatrix} X_{new} \\ Y_{new} \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} X_{old} \\ Y_{old} \\ 1 \end{bmatrix}
$$

Applying transformations on equilateral triangle:

Consider that coordinates of vertices of equilateral triangle are (X1,Y1), (X2,Y2) and (X3,Y3). After applying basic transformations, we will get corresponding coordinates as (X1',Y1'), (X2',Y2') and (X3',Y3') respectively. Following multiplication will give the translation on this equilateral triangle:

$$\begin{bmatrix} X1' & X2' & X3' \\ Y1' & Y2' & Y3' \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & T_x \\ 0 & 1 & T_y \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} X1 & X2 & X3 \\ Y1 & Y2 & Y3 \\ 1 & 1 & 1 \end{bmatrix}$$

Similarly we can apply rotation and scaling on equilateral triangle.

Applying transformations on rhombus:

Consider that coordinates of vertices of rhombus are (X1,Y1), (X2,Y2), (X3,Y3) and (X4,Y4) applying basic transformations, we will get corresponding coordinates as (X1',Y1'), (X2',Y2'), (X3',Y3') and (X4',Y4') respectively. Following multiplication will give the translation on this

rhombus:

$$\begin{bmatrix} X1' & X2' & X3' & X4' \\ Y1' & Y2' & Y3' & Y4' \\ 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & T_x \\ 0 & 1 & T_y \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} X1 & X2 & X3 & X4 \\ Y1 & Y2 & Y3 & Y4 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Similarly we can apply rotation and scaling on rhombus.

Input: Enter number of vertices

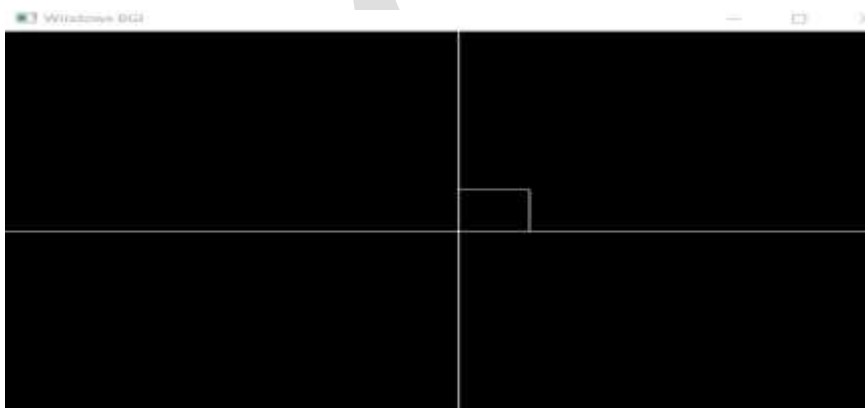Enter (x,y) Co-ordinate of point P

Enter X-Translation tx

Enter X-Translation ty

Polygon after Translation

Enter the angle of rotation in degrees

Enter your choice

Output:

Conclusion: Hence, we have studied that 2-D object and perform following basic transformations: Scaling, Translation, Rotation. Apply the concept of operator overloading.

Questions:

1. How to rotate any 2-D object about an arbitrary point? Explain in brief.
2. Explain the concept of operator overloading with example.

Assignment No.	5 (GROUP B)
Title	Curves and fractals
Subject	Computer Graphics Laboratory
Class	S.E. (C.E.)
Roll No.	
Date	
Signature	

Assignment No. 5

Title: Curves and fractals

Problem Statement: a) Write C++ program to generate snowflake using concept of fractals.
OR
b) Write C++ program to generate Hilbert curve using concept of fractals.
OR
c) Write C++ program to generate fractal patterns by using Koch curves..

Prerequisites: Basic programming skills of C++

64-bit Open source Linux

Open Source C++ Programming tool like G++/GCC

**Objectives:** To study curves and fractals.

Theory:

Koch Curve:

The Koch curve fractal was first introduced in 1904 by Helge von Koch. It was one of the firstfractal objects to be described. To create a Koch curve
1. Create a line and divide it into three parts.
2. The second part is now rotated by 60°.
3. Add another part which goes from the end of part 2 to the beginning of part 3
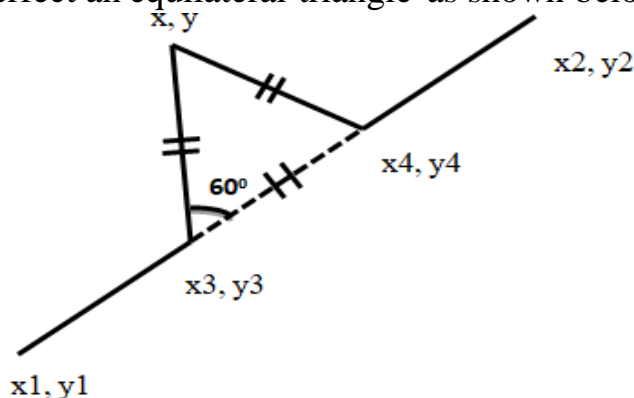4. Repeat step 1 to step 3 with each part of the line.
We will get following Koch curve as number of iteration goes on increasing
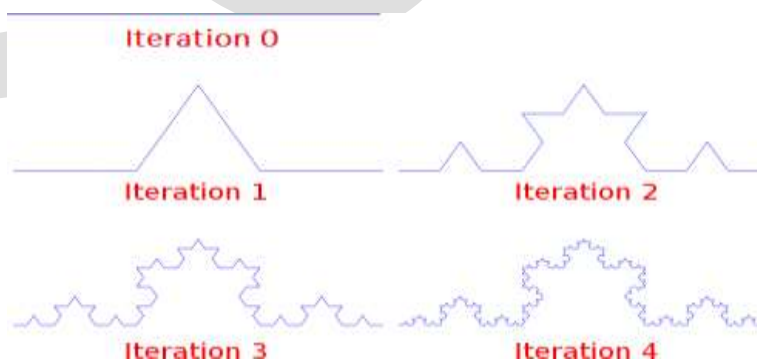
Step 1: In Iteration 0, we have a horizontal line.
Step 2: In Iteration 1, line is divided into 3 equal parts. Middle part of a line is rotated in

60^0,because it forms a perfect an equilateral triangle as shown below:

Here, (x1,y1) and (x2, y2) is accepted from user.
Now, we can see line is divided into 3 equal segments: segment((x1,y1),(x3,y3)), segment((x3,y3),(x4,y4)),segment((x4,y4),(x2,y2)) in above figure.

Coordinates of middle two points will be calculated as follows:
x3 = (2*x1+x2)/3;
      y3 = (2*y1+y2)/3;x4 =
(x1+2*x2)/3;
      y4 = (y1+2*y2)/3;

In our curve, middle segment((x3,y3),(x4,y4)) will not be drawn. Now, in order to find out coordinates of the top vertex (x,y) of equilateral triangle, we have rotate point (x4,y4) with respect to arbitrary point (x3,y3) by angle of 60 degree in anticlockwise direction. After performing this rotation, we will get rotated coordinates (x, y) as:

$$x = x3 + (x4 - x3) * \cos \theta + (y4 - y3) * \sin \theta$$
$$y = y3 - (x4 - x3) * \sin \theta + (y4 - y3) * \cos\theta$$

Step 3: In iteration 2, you will repeat step 2 for every segment obtained in iteration1.In this way, you can generate Koch curve for any number of iterations.
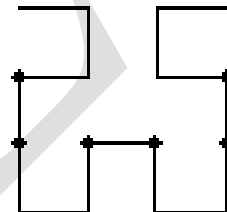
The Hilbert curve

The Hilbert curve is a space filling curve that visits every point in a square grid with a size of 2×2, 4×4, 8×8, 16×16, or any other power of 2. It was first described by David Hilbert in 1892. Applications of the Hilbert curve are in image processing: especially image compression and dithering. It has advantages in those operations where the coherence

between neighbouring pixels isimportant. The Hilbert curve is also a special version of a quad tree; any image processing function that benefits from the use of quad trees may also use a Hilbert curve.

Cups and joins

The basic elements of the Hilbert curves are what I call "cups" (a square with one open side) and "joins" (a vector that joins two cups). The "open" side of a cup can be top, bottom, left or right. In addition, every cup has two end-points, and each of these can be the "entry" point or the "exit" point.So, there are eight possible varieties of cups. In practice, a Hilbert curve uses only four types of cups. In a similar vein, a join has a direction: up, down, left or right.

A first order Hilbert curve is just a single cup (see the figure on theleft). It fills a 2×2 space. The second order Hilbert curve replaces that cup by four (smaller) cups, which are linked together by three joins (see the figure on the right; the link between a cup and a joinhas been marked with a fat dot in the figure). Every next order repeats the process or replacing each cup by four smaller cups and three joins.
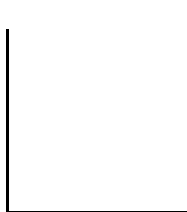
Cup subdivision rules

$$\sqcup \Rightarrow \sqsupset \downarrow \sqcup \rightarrow \sqcup \uparrow \sqsubset$$

$$\sqsupset \Rightarrow \sqcup \rightarrow \sqsupset \downarrow \sqsupset \leftarrow \sqcap$$

$$\sqcap \Rightarrow \sqsubset \uparrow \sqcap \leftarrow \sqcap \downarrow \sqsupset$$

$$\sqsubset \Rightarrow \sqcap \leftarrow \sqsubset \uparrow \sqsubset \rightarrow \sqcup$$

The function presented below (in the "C" language) computes the Hilbert curve. Note that the curve is symmetrical around the vertical axis. It would therefore be sufficient to draw half of the Hilbert curve.

Snowflake curve:

Snowflake curve is drawn using koch curve iterations. In koch curve, we just have a single line in thestarting iteration and in snowflake curve, we have an equilateral triangle. Draw an equilateral triangleand repeat the steps of Koch curve generation for all three segments of an equilateral triangle.

Iteration 0

Iteration 1

Iteration 2

Iteration 3
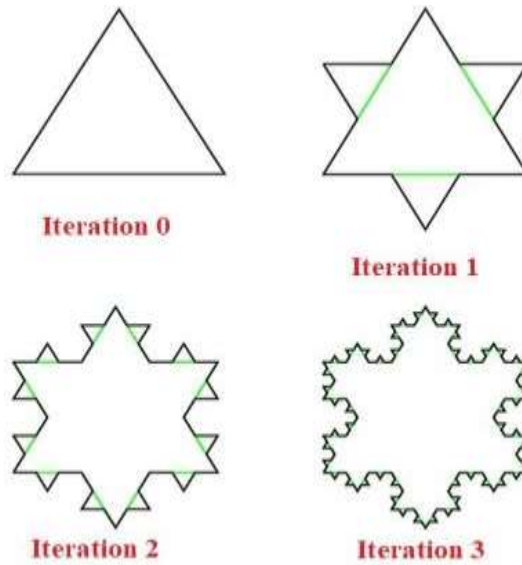
Input: Give the value of n:

x3 = (2*x1+x2)/3;
y3 = (2*y1+y2)/3;
x4 = (x1+2*x2)/3;
y4 = (y1+2*y2)/3;

Output:
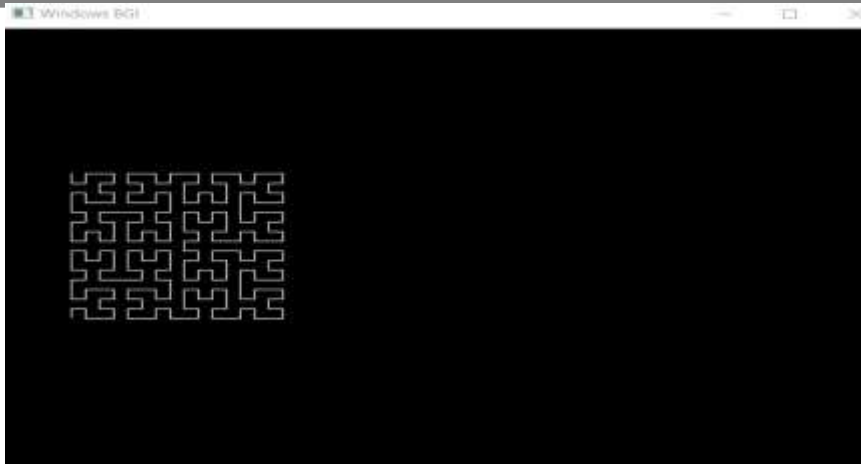
Conclusion: Hence, we have studied that to generate Hilbert curve using concept of fractals.

Questions:

1. What is the importance of curves and fractals in computer graphics?
2. What are applications of curves and fractals?

GROUP C

Assignment No.	6 (GROUP C)
Title	Implementation of OpenGL functions
Subject	Computer Graphics Laboratory
Class	S.E. (C.E.)
Roll No.	
Date	
Signature	

Assignment No. 6

Title: Implementation of OpenGL functions

Problem Statement: a) Design and simulate any data structure like stack or queue visualization using graphics. Simulation should include all operations performed ondesigned data structure. Implement the same using OpenGL.

OR

b) Write C++ program to draw 3-D cube and perform following transformations on it using OpenGL i) Scaling ii) Translation iii) Rotation about an axis (X/Y/Z).

OR

C) Write OpenGL program to draw Sun Rise and Sunset.

Prerequisites: Basic programming skills of C++

64-bit Open source Linux

Open Source C++ Programming tool like G++/GCC

Objectives: To implement OpenGL functions.

Theory:
OpenGL Basics:

Open Graphics Library (OpenGL) is a cross-language (language independent), cross-platform (platform independent) API for rendering 2D and 3D Vector Graphics (use of polygons to represent image). OpenGL is a low-level, widely supported modeling and rendering software package, availableacross all platforms. It can be used in a range of graphics applications, such as games, CAD design, or modeling. OpenGL API is designed mostly in hardware.

- **Design**: This API is defined as a set of functions which may be called by the client program. Although functions are similar to those of C language but it is language independent.

- **Development**: It is an evolving API and Khronos Group regularly releases its new version having some extended feature compare to previous one. GPU vendors may also provide some additional functionality in the form of extension.

- **Associated Libraries:** The earliest version is released with a companion library called OpenGL utility library. But since OpenGL is quite a complex process. So in order to make it easier other library such as OpenGL Utility Toolkit is added which is later superseded by freeglut. Later included library were GLEE, GLEW and glbinding.

- **Implementation:** Mesa 3D is an open source implementation of OpenGL. It can do pure software rendering and it may also use hardware acceleration on BSD, Linux, and other platforms bytaking advantage of Direct Rendering Infrastructure.

Installation of OpenGL on Ubuntu

We need the following sets of libraries in programming OpenGL:

1. **Core OpenGL (GL):** consists of hundreds of functions, which begin with a prefix "gl" (e.g.,glColor, glVertex, glTranslate, glRotate). The Core OpenGL models an object via a set of geometricprimitives, such as point, line, and polygon.

2. OpenGL Utility Library (GLU): built on-top of the core OpenGL to provide important utilities and more building models (such as qradric surfaces). GLU functions start with a prefix "glu"(e.g.,

 gluLookAt, gluPerspective).

3. OpenGL Utilities Toolkit (GLUT): provides support to interact with the Operating System (such as creating   a window, handling key and mouse inputs); and more building models (such as sphere and torus). GLUT functions start with a prefix of "glut" (e.g., glutCreatewindow, glutMouseFunc). GLUT is designed for constructing small to medium

sized OpenGL programs. While GLUT is well-suited to learning OpenGL and developing simple OpenGL applications, GLUT is not a full-featured toolkit so large applications requiring sophisticateduser interfaces are better off using native window system toolkits. GLUT is simple, easy, and small.
Alternative of GLUT includes SDL.

4. OpenGL Extension Wrangler Library (GLEW): "GLEW is a cross-platform open-

sourceC/C++ extension loading library. GLEW provides efficient run-time mechanisms for

determining which OpenGL extensions are supported on the target platform.

- sudo apt-get install freeglut3-dev

For working on Ubuntu operating system:
- gcc filename.c -lGL -lGLU -lglut
where filename.c is the name of the file with which this program is saved.

Prerequisites for OpenGL

Since OpenGL is a graphics API and not a platform of its own, it requires a language to operate inand the language of choice is C++.

Getting Started with OpenGL:

Overview of an OpenGL Program:

- Main
- ✓ Open window and configure frame buffer (using GLUT for example)
- ✓ Initialize GL states and display (Double buffer, color mode, etc.)
- Loop
    - ✓ Check for events

if window event (resize, unhide, maximize etc.)modify the view report and Redraw else if input event(keyboard and mouse etc) handle the event (such as move the camera or change the state) *and usually draw the scene*

- Redraw
- ✓ Clear the screen (and buffers e.g., z-buffer)
- ✓ Change states (if desired)
- ✓ Render
- ✓ Swap buffers (if double buffer)

OpenGL order of operations
- Construct shapes (geometric descriptions of objects – vertices, edges, polygons etc.)
- Use OpenGL to
o   Arrange shape in 3D (using transformations)
o   Select your vantage point (and perhaps lights)

o   Calculate color and texture properties of each object
o   Convert shapes into pixels on screen

OpenGL Syntax
- All functions have the form: gl*

o   *glVertex3f()* – 3 means that this function take three arguments, and f means that the type ofthose arguments is float.

o   *glVertex2i()* – 2 means that this function take two arguments, and i means that the type ofthose arguments is integer
☐   All variable types have the form: GL*
o   In OpenGL program it is better to use OpenGL variable types (portability)
▪   *Glfloat* instead of *float*
▪   *Glint* instead of *int*

OpenGL primitives

Drawing two lines
glBegin(GL_LINES);

*glVertex3f(_,_,_); // start point of line 1glVertex3f(_,_,_); //
end point of line 1 glVertex3f(_,_,_); // start point of line 2
glVertex3f(_,_,_); // end point of line 2 glEnd();*

We can replace GL_*LINES* with *GL_POINTS, GL_LINELOOP, GL_POLYGON* etc.

OpenGL states
- On/off (e.g., depth buffer test)
o *glEnable(GLenum)*
o *glDisable(GLenum)*
o Examples:
▪ *glEnable(GL_DEPTH_TEST);*
▪ *glDisable(GL_LIGHTING);*
- Mode States
o Once the mode is set the effect stays until reset
o Examples:
▪ *glShadeModel(GL_FLAT) or glShadeModel(GL_SMOOTH)*
▪ *glLightModel(…)* etc.

Drawing in 3D
- Depth buffer (or z-buffer) allows scene to remove hidden surfaces. Use *glEnable(GL_DEPTH_TEST)* to enable it.
o glPolygonMode(Face, Mode)
- Face: GL_FRONT, GL_BACK, GL_FRONT_AND_BACKMode: GL_LINE, GL_POINT, GL_FILL
o glCullFace(Mode)
  ✓ Mode: GL_FRONT, GL_BACK, GL_FRONT_AND_BACK
  ✓ glFrontFace(Vertex_Ordering)
  ✓ Vertex Ordering: GL_CW or GL_CCW

Viewing transformation
☐ glMatrixMode (Mode)
  ✓ Mode: GL_MODELVIEW, GL_PROJECTION, or GL_TEXTURE
  ✓ glLoadIdentity()
  ✓ glTranslate3f(x,y,z)
  ✓ glRotate3f(angle,x,y,z)
  ✓ glScale3f(x,y,z)

OpenGL provides a consistent interface to the underlying graphics hardware. This abstraction allows a single program to run a different graphics hardware easily. A program written with OpenGL can even be run in software (slowly) on machines with no graphics acceleration. OpenGL function names always begin with *gl*, such as *glClear(),* and they may end with characters that indicate the types of the parameters, for example *glColor3f(GLfloat red, GLfloat green, GLfloat blue)* takes three floating-point color parameters and *glColor4dv(const GLdouble *v)* takes a pointer to an array that contains 4 double-precision floating-point values. OpenGL constants begin with *GL*, such as *GL DEPTH*. OpenGL also uses special names for types that are passed to its functions, such as *GLfloat* or *GLint*, the corresponding C types are compatible, that is *float* and *int* respectively.

GLU is the OpenGL utility library. It contains useful functions at a higher level than those provided by OpenGL, for example, to draw complex shapes or set up cameras. All GLU functions are written on top of OpenGL. Like OpenGL, GLU function names begin with glu, and constants begin with GLU.

GLUT, the OpenGL Utility Toolkit, provides a system for setting up callbacks for interacting with the user and functions for dealing with the windowing system. This abstraction allows a program to run on different operating systems with only a recompile. Glut follows the convention of prepending function names with glut and constants with GLUT.

Writing an OpenGL Program with GLUT

An OpenGL program using the three libraries listed above must include the appropriate headers. This requires the following three lines:
#include <GL/gl.h>

#include <GL/glu.h>
#include <GL/glut.h>

Before OpenGL rendering calls can be made, some initialization has to be done. With GLUT, this consists of initializing the GLUT library, initializing the display mode, creating the window, and setting up callback functions. The following lines initialize a full color, double buffered display: *glutInit(&argc, argv);*
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);

Double buffering means that there are two buffers, a front buffer and a back buffer. The front buffer is displayed to the user, while the back buffer is used for rendering operations. This prevents flickering that would occur if we rendered directly to the front buffer.

Next, a window is created with GLUT that will contain the viewport which displays the OpenGL front buffer with the following three lines:

glutInitWindowPosition(px, py);
glutInitWindowSize(sx, sy);
glutCreateWindow(name);

To register callback functions, we simply pass the name of the function that handles the event to theappropriate GLUT function.
glutReshapeFunc(reshape);
glutDisplayFunc(display);

Here, the functions should have the following prototypes:
void reshape(int width, int height);
void display();

In this example, when the user resizes the window, reshape is called by GLUT, and when the displayneeds to be refreshed, the display function is called. For animation, an idle event handler that takes no arguments can be created to call the display function to constantly redraw the scene with *glutIdleFunc*. Once all the callbacks have been set up, a call to *glutMainLoop* allows the program to run.
In the display function, typically the image buffer is cleared, primitives are rendered to it, and the results are presented to the user. The following line clears the image buffer, setting each pixel color to the clear color, which can be configured to be any color:
glClear(GL_COLOR_BUFFER_BIT);

The next line sets the current rendering color to blue. OpenGL behaves like a state machine, so certainstate such as the rendering color is saved by OpenGL and used automatically later as it is needed.
glColor3f(0.0f, 0.0f, 1.0f);

To render a primitive, such as a point, line, or polygon, OpenGL requires that a call to *glBegin* is made to specify the type of primitive being rendered.
glBegin(GL_LINES);

Only a subset of OpenGL commands is available after a call to *glBegin*. The main command that is used is *glVertex*, which specifies a vertex position. In GL LINES mode, each pair of vertices define endpoints of a line segment. In this case, a line would be drawn from the point at (x0, y0) to (x1, y1).
glVertex2f(x0, y0); glVertex2f(x1, y1);

A call to glEnd completes rendering of the current primitive. *glEnd();* Finally, the back buffer needsto be swapped to the front buffer that the user will see, which GLUT can handle for us:

glutSwapBuffers();

Developer-Driven Advantages

- ### Industry standard

An independent consortium, the OpenGL Architecture Review Board, guides the OpenGL specification. With broad industry support, OpenGL is the only truly open, vendor-neutral, multiplatform graphics standard.

- ### Stable

OpenGL implementations have been available for more than seven years on a wide variety of platforms. Additions to the specification are well controlled, and proposed updates are announced in time for developers to adopt changes. Backward compatibility requirements ensure that existing applications do not become obsolete.

- ### Reliable and portable

All OpenGL applications produce consistent visual display results on any OpenGL API-compliant hardware, regardless of operating system or windowing system.

- ### Evolving

Because of its thorough and forward-looking design, OpenGL allows new hardware innovations to be accessible through the API via the OpenGL extension mechanism. In this way, innovations appear in the API in a timely fashion, letting application developers and hardware vendors incorporate new features into their normal product release cycles.

- ### Scalable

OpenGL API-based applications can run on systems ranging from consumer electronics to PCs, workstations, and supercomputers. As a result, applications can scale to any class of machine that the developer chooses to target.

- ### Easy to use

OpenGL is well structured with an intuitive design and logical commands. Efficient OpenGL routines typically result in applications with fewer lines of code than those that make up programs generated using other graphics libraries or packages. In addition, OpenGL drivers encapsulate information about the underlying hardware, freeing the application developer from having to design for specific hardware features.

- **Well-documented:**

Numerous books have been published about OpenGL, and a great deal of sample code is readily available, making information about OpenGL inexpensive and easy to obtain.

Input:

    glTranslate3f(x,y,z)
    glRotate3f(angle,x,y,z)
    glScale3f(x,y,z)

Output:
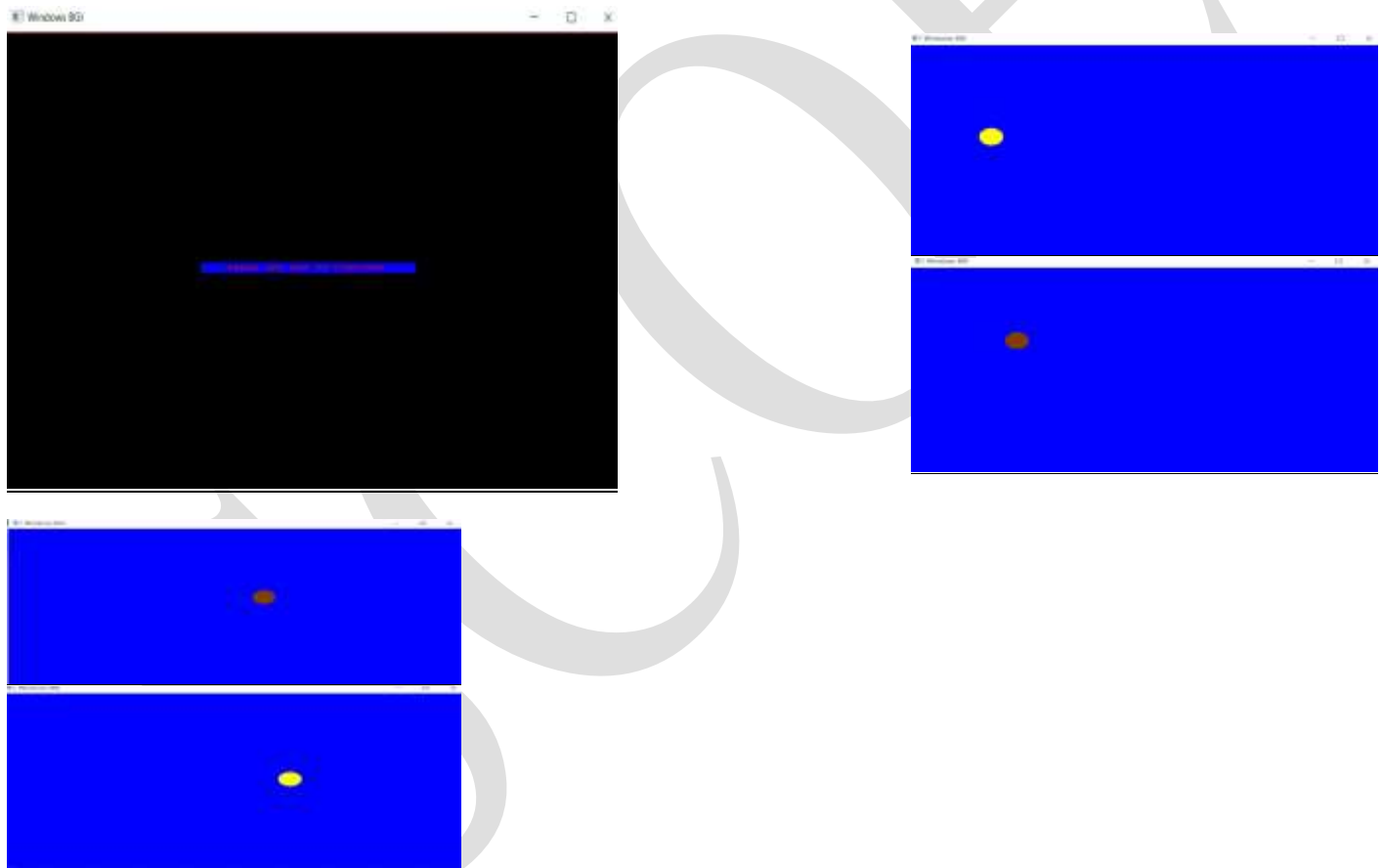
Conclusion: Hence, we have studied that OpenGL program to draw Sun Rise and Sunset.

Questions:
1.What is OpenGL Program Explain in details?
2.what are advantages of Developer-driven ?

Assignment No.	7 (GROUP C)
Title	Animation using C++
Subject	Computer Graphics Laboratory
Class	S.E. (C.E.)
Roll No.	
Date	
Signature	

Assignment No. 7

Title: Animation using C++

Problem Statement: a) Write a C++ program to control a ball using arrow keys. Apply the concept of polymorphism.

<div align="center">

OR

</div>

b) Write a C++ program to implement bouncing ball using sine wave form. Apply the concept of polymorphism.

<div align="center">

OR

</div>

c) Write C++ program to draw man walking in the rain with an umbrella. Apply the concept of polymorphism.

<div align="center">

OR

</div>

Write a C++ program to implement the game of 8 puzzle. Apply the concept of polymorphism.

<div align="center">

OR

</div>

d) Write a C++ program to implement the game Tic Tac Toe. Apply the concept of polymorphism.

Prerequisites: Basic programming skills of C++

64-bit Open source Linux
Open Source C++ Programming tool like G++/GCC

Objectives: To learn scanline polygon fill algorithm

Theory:
 What is animation?

 Animation is the process of designing, drawing, making layouts and preparation of photographic sequences which are integrated in the multimedia and gaming products. Animation involves the exploitation and management of still images to generate the illusion of movement. How to move an element to left, right, up and down using arrow keys?
To detect which arrow key is pressed, you can use ncurses.h header file. Arrow key's code is defined as: KEY_UP, KEY_DOWN, KEY_LEFT, KEY_RIGHT.
#include<ncurses.h>int main()
{
int ch;

```
/* Curses Initialisations */initscr();
raw();
keypad(stdscr, TRUE);noecho();

printw("Welcome - Press # to Exit\n");while((ch =

getch()) != '#')
{
switch(ch)
{
case KEY_UP: printw("\nUp Arrow");

break;
case KEY_DOWN: printw("\nDown Arrow");break;
case KEY_LEFT: printw("\nLeft Arrow");break;
case KEY_RIGHT: printw("\nRight Arrow");break;
default:
{
printw("\nThe pressed key is ");
attron(A_BOLD);
printw("%c", ch);
attroff(A_BOLD);
}
}
}

printw("\n\nBye Now!\n");refresh();
getch();
endwin();
return 0;
}
```

How to draw a sine wave using c++?

```
#include <math.h> #include
<graphics.h> #include <iostream>

int main() {
     int gd = DETECT, gm;int angle
     = 0; double x, y;
```

```
    initgraph(&gd, &gm, NULL);

    line(0, getmaxy() / 2, getmaxx(), getmaxy() / 2);
    /* generate a sine wave */
    for(x = 0; x < getmaxx(); x+=3) {

    /* calculate y value given x */y =
    50*sin(angle*3.141/180);
    y = getmaxy()/2 - y;
    /* color a pixel at the given position */putpixel(x, y, 15);
    delay(100);
  /* increment angle */angle+=5;
 }
 /* deallocate memory allocated for graphics screen */closegraph();
 return 0;
}
```

A game of 8 puzzle:

      An 8 puzzle is a simple game consisting of a 3 x 3 grid (containing 9 squares). One of the squares is empty. The object is to move to squares around into different positions and having the numbers displayed in the "goal state". The image to the left can be thought of as an unsolved initial state of the "3 x 3" 8 puzzle. Eight digits will in random order. To solve a puzzle, you have to move blocks by performing translation of blocks.

Initial State   **Goal State**
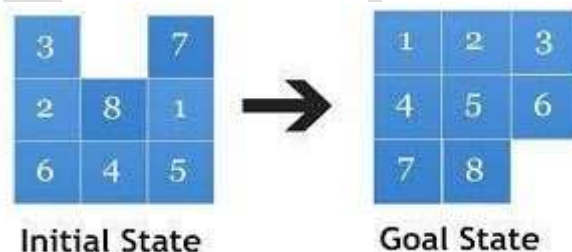
## Implementation of Tic-Tac-Toe game
Rules of the Game

•   The game is to be played between two people (in this program between HUMAN and COMPUTER).

•   One of the player chooses 'O' and the other 'X' to mark their respective cells.

•   The game starts with one of the players and the game ends when one of the players has one whole row/ column/ diagonal filled with his/her respective character ('O' or 'X').

•   If no one wins, then the game is said to be a draw.

Note: In all above programs, you have to perform translation of an image to show animation effect.

Input:

Initial State   Goal State
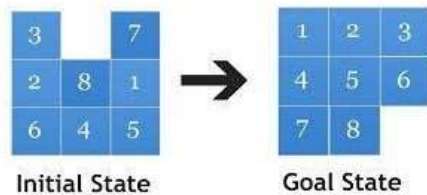
Output:

Conclusion: Hence, we have studied that bouncing ball using sine wave form. Apply the concept of polymorphism.

Questions:

1. What is Animation explain in details?
2. What is an 8 Puzzle Program? Explain in details?

Assignment No.	(Mini Project Case Study)
Title	Design and implement game / animation clip / Graphics Editor using open source graphics library. Make use of maximum features of Object Oriented Programming
Subject	Computer Graphics Laboratory
Class	S.E. (C.E.)
Roll No.	
Date	
Signature	