Program 1

/*

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

*/

```
# include<iostream>
using namespace std;
class Complex                    //decaring Class Complex
{
 double real;
 double img;
 public:
 Complex();          // Default Constructor
 friend istream & operator >> (istream &, Complex &); // Input
 friend ostream & operator << (ostream &, const Complex &); // Output
 Complex operator + (Complex); // Addition
 Complex operator * (Complex); // Multiplication
 };
 Complex::Complex()              // Default Constructor
 {
 real = 0;
 img = 0;
 }
 istream & operator >> (istream &, Complex & i)
 {
```

```
cin >> i.real >> i.img;
return cin;
}
ostream & operator << (ostream &, const Complex & d)
{
cout << d.real << " + " << d.img << "i" << endl;
return cout;
}
Complex Complex::operator + (Complex c1)   // Overloading + operator
{
Complex temp;
temp.real = real + c1.real;
temp.img = img + c1.img;
return temp;
}
Complex Complex::operator * (Complex c2)   // Overloading * Operator
{
Complex tmp;
tmp.real = real * c2.real - img * c2.img;
tmp.img = real * c2.img + img * c2.real;
return tmp;
}
int main()
{
Complex C1, C2, C3, C4;
int flag = 1;
char b;
while (flag == 1)
{
cout << "Enter Real and Imaginary part of the Complex Number 1 : \n";
cin >> C1;
```

```
cout << "Enter Real and Imaginary part of the Complex Number 2 : \n";
cin >> C2;
int f = 1;
while (f == 1)
{
cout << "Complex Number 1 : " << C1 << endl;
cout << "Complex Number 2 : " << C2 << endl;
cout << "**********MENU**********" << endl;
cout << "1. Addition of Complex Numbers" << endl;
cout << "2. Multiplication of Complex Numbers" << endl;
cout << "3. Exit\n";
int a;
cout << "Enter your choice from above MENU (1 to 3) : ";
cin >> a;
if (a == 1)
{
C3 = C1+C2;
cout << "Addition : " << C3 << endl;
cout << "Do you wan to perform another operation (y/n) : \n";
cin >> b;
if (b == 'y' | | b == 'Y')
{
f=1;
}
else
{
cout << "Thanks for using this program!!\n";
flag=0;
f=0;
}
}
```

```
else if (a == 2)
{
C4 = C1 * C2;
cout << "Multiplication : " << C4 << endl;
cout << "Do you wan to perform another operation (y/n) : \n";
cin >> b;
if (b == 'y' | | b == 'Y')
{
f=1;
}
else
{
cout << "Thanks for using this program!!\n";
flag=0;
f=0;
}
}
else
{
cout << "Thanks for using this program!!\n";
flag=0;
f=0;
}
}
}
return 0;
}
```

Program 2

/*

Experiment Number 2 : Develop a program in C++ to create a database of student's information system

containing the following information: Name, Roll number, Class, Division,

Date of Birth, Blood group,

Contactaddress, Telephone number, Driving license no. and other. Construct

the database with

suitable member functions. Make use of constructor, default constructor,

copy constructor,

destructor, static member functions, friend class, this pointer, inline

code and dynamic

memory allocation operators-new and delete as well as exception handling.

***/**

```
#include<iostream>
#include<string.h>
using namespace std;

class StudData;

class Student{
   string name;
   int roll_no;
   string cls;
   char* division;
   string dob;
   char* bloodgroup;
   static int count;

   public:

   Student()        // Default Constructor
```

```
    {
        name="";
        roll_no=0;
        cls="";
        division=new char;
        dob="dd/mm/yyyy";
        bloodgroup=new char[4];
    }

    ~Student()
    {
        delete division;
        delete[] bloodgroup;
    }

    static int getCount()
    {
        return count;
    }

    void getData(StudData*);
    void dispData(StudData*);
};

class StudData{
    string caddress;
    long int* telno;
    long int* dlno;
    friend class Student;

    public:
```

```
StudData()
{
   caddress="";
   telno=new long;
   dlno=new long;
}

~StudData()
{
   delete telno;
   delete dlno;
}

void getStudData()
{
   cout<<"Enter Contact Address : ";
   cin.get();
   getline(cin,caddress);
   cout<<"Enter Telephone Number : ";
   cin>>*telno;
   cout<<"Enter Driving License Number : ";
   cin>>*dlno;
}

void dispStudData()
{
   cout<<"Contact Address : "<<caddress<<endl;
   cout<<"Telephone Number : "<<*telno<<endl;
   cout<<"Driving License Number : "<<*dlno<<endl;
}
```

```
};

inline void Student::getData(StudData* st)
{
   cout<<"Enter Student Name : ";
   getline(cin,name);
   cout<<"Enter Roll Number : ";
   cin>>roll_no;
   cout<<"Enter Class : ";
   cin.get();
   getline(cin,cls);
   cout<<"Enter Division : ";
   cin>>division;
   cout<<"Enter Date of Birth : ";
   cin.get();
   getline(cin,dob);
   cout<<"Enter Blood Group : ";
   cin>>bloodgroup;
   st->getStudData();
   count++;
}

inline void Student::dispData(StudData* st1)
{
   cout<<"Student Name : "<<name<<endl;
   cout<<"Roll Number : "<<roll_no<<endl;
   cout<<"Class : "<<cls<<endl;
   cout<<"Division : "<<division<<endl;
   cout<<"Date of Birth : "<<dob<<endl;
   cout<<"Blood Group : "<<bloodgroup<<endl;
   st1->dispStudData();
```

```
}

int Student::count;

int main()
{
   Student* stud1[100];
   StudData* stud2[100];
   int n=0;
   char ch;

   do
   {
      stud1[n]=new Student;
      stud2[n]=new StudData;
      stud1[n]->getData(stud2[n]);
      n++;
      cout<<"Do you want to add another student (y/n) : ";
      cin>>ch;
      cin.get();
   } while (ch=='y' || ch=='Y');

   for(int i=0;i<n;i++)
   {
      cout<<"-----------------------------------------------------------------"<<endl;
      stud1[i]->dispData(stud2[i]);
   }

   cout<<"-----------------------------------------------------------------"<<endl;
   cout<<"Total Students : "<<Student::getCount();
   cout<<endl<<"-----------------------------------------------------------------"<<endl;
```

```
   for(int i=0;i<n;i++)
   {
      delete stud1[i];
      delete stud2[i];
   }


   return 0;
}
```

Program 3

```
/*
Imagine a publishing company which does marketing for book
and audiocassette versions.
 Create a class publication that stores the title (a string)
and price (type float) of a
 publication.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 classes,
allows user to enter data and
 displays the data members.If an exception is caught, replace
all the data member values
 with zero values.
*/

# include<iostream>
# include<stdio.h>
using namespace std;
```

```
class publication               // declaring class Publication
 {
 private:
 string title;
 float price;
 public:
 void add()
 {
 cout << "\nEnter the Publication information : " << endl;
 cout << "Enter Title of the Publication : ";
cin.ignore();
getline(cin, title);
cout << "Enter Price of Publication : ";
 cin >> price;
 }
 void display()
 {
 cout << "\n-----------------------------------------------";
 cout << "\nTitle of Publication : " << title;
 cout << "\nPublication Price : " << price;
 }
 };
 class book : public publication  // declaring class book which inherits class publication in public mode.
 {
 private:
 int page_count;
 public:
 void add_book()
 {
 try
```

```
{
add();
cout << "Enter Page Count of Book : ";
cin >> page_count;
if (page_count <= 0)
{
throw page_count;
}
}
catch(...)
{
cout << "\nInvalid Page Count!!!";
page_count = 0;
}
}
void display_book()
{
display();
cout << "\nPage Count : " <<
page_count;
cout << "\n-------------------------------------------------\n";
}
};
class tape : public publication     // declaring class tape which inherits class publication in public
mode
{
private:
float play_time;
public:
void add_tape()
{
```

```
try
{
add();
cout << "Enter Play Duration of the Tape : ";
cin >> play_time;
if (play_time <= 0)
throw play_time;
}
catch(...)
{
cout << "\nInvalid Play Time!!!";
play_time = 0;
}
}
void display_tape()
{
display();
cout << "\nPlay Time : " <<
play_time << " min";
cout << "\n-------------------------------------------------\n";
}
};
int main()
{
book b1[10];          // object of class book
tape t1[10];          // object of class tape
int ch, b_count = 0, t_count = 0;
do
{
cout << "\n* * * * * PUBLICATION DATABASE SYSTEM * * * * *";
cout << "\n--------------------MENU----------------------";
```

```
cout << "\n1. Add Information to Books";
cout << "\n2. Add Information to Tapes";
cout << "\n3. Display Books Information";
cout << "\n4. Display Tapes Information";
cout << "\n5. Exit";
cout << "\n\nEnter your choice : ";
cin >> ch;
switch(ch)
{
case 1:
b1[b_count].add_book();
b_count + +;
break;
case 2:
t1[t_count].add_tape();
t_count + +;
break;
case 3:
cout << "\n* * * * BOOK PUBLICATION DATABASE SYSTEM * * * *";
for (int j=0;j < b_count;j++)
{
b1[j].display_book();
}
break;
case 4:
cout << "\n* * * * TAPE PUBLICATION DATABASE SYSTEM * * * *";
for (int j=0;j < t_count;j++)
{
t1[j].display_tape();
}
break;
```

```
 case 5:
 exit(0);
 }
 }while (ch != 5);
 return 0;
 }
```

Program 4

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


#include<iostream>
#include<fstream>
using namespace std;
class Employee            // declaring class employee
{
 string Name;
 int ID;
 double salary;
 public:
 void accept()
 {
 cout<<"\n Name : ";
 cin.ignore();
getline(cin,Name);
 cout<<"\n Id : ";
 cin>>ID;
 cout<<"\n Salary : ";
 cin>>salary;
 }
```

```
 void display()
 {
 cout<<"\n Name : "<<Name;
 cout<<"\n Id : "<<ID;
 cout<<"\n Salary : "<<salary<<endl;
 }
};

int main()
{
 Employee o[5];
 fstream f;
 int i,n;

 f.open("demo.txt",ios::out);
 cout<<"\n Enter the number of employees you want to store : ";
 cin>>n;
 for(i=0;i<n;i++)
 {
 cout<<"\n Enter information of Employee "<<i+1<<"\n";
 o[i].accept();
 f.write((char*)&o[i],sizeof o[i]);
 }

 f.close();

 f.open("demo.txt",ios::in);
 cout<<"\n Information of Employees is as follows : \n";
 for(i=0;i<n;i++)
 {
 cout<<"\nEmployee "<<i+1<<"\n";
```

```
 f.write((char*)&o[i],sizeof o[i]);
 o[i].display();
 }
 f.close();

 return 0;
}
```

Program 5

```
/*
Write a function template for selection sort that inputs, sorts and outputs an integer array and
a float array.
*/

#include<iostream>
using namespace std;
int n;
#define size 10
template<class T>
void sel(T A[size])
{
   int i,j,min;
   T temp;
   for(i=0;i<n-1;i++)
   {
      min=i;
      for(j=i+1;j<n;j++)
      {
         if(A[j]<A[min])
         min=j;
      }
      temp=A[i];
```

```
      A[i]=A[min];
      A[min]=temp;
   }
   cout<<"\nSorted array:";
   for(i=0;i<n;i++)
   {
      cout<<" "<<A[i];
   }
}

int main()
{
   int A[size];
   float B[size];
   int i;
   int ch;
   do
         {
                  cout<<"\n* * * * * SELECTION SORT SYSTEM * * * * *";
                  cout<<"\n--------------------MENU----------------------";
                  cout<<"\n1. Integer Values";
                  cout<<"\n2. Float Values";
                  cout<<"\n3. Exit";
                  cout<<"\n\nEnter your choice : ";
                  cin>>ch;

                  switch(ch)
                  {
                           case 1:
                                    cout<<"\nEnter total no of int elements:";
            cin>>n;
```

```
            cout<<"\nEnter int elements:";
            for(i=0;i<n;i++)
            {
               cin>>A[i];
            }
            sel(A);
                                    break;

                        case 2:
                                    cout<<"\nEnter total no of float elements:";
            cin>>n;
            cout<<"\nEnter float elements:";
            for(i=0;i<n;i++)
            {
               cin>>B[i];
            }
            sel(B);
                                    break;

                        case 3:
                                    exit(0);
                  }
           }while(ch!=3);




  return 0;

}
```

Program 6

```
/*
```

```
Write C++ Program using STL for sorting and searching user defined
records such as item records using vector container.
*/
#include <iostream> //standard input output stream header file
#include <algorithm> //The STL algorithms are generic because
they can operate on a variety of data structures
#include <vector> //The header file for the STL vector library is
vector.
using namespace std;
class Item // creating class Item
{
 public:
 char name[10];
 int quantity;
 int cost;
 int code;
 bool operator==(const Item& i1) //Boolean operators allow
you to create more complex conditional statements
 {
 if(code==i1.code) //operator will return 1 if the
comparison is true, or 0 if the comparison is false
 return 1;
 return 0;
 }
 bool operator<(const Item& i1)
 {
 if(code<i1.code) //operator will return 1 if the
comparison is true, or 0 if the comparison is false
 return 1;
 return 0;
 }
```

```
};
vector<Item> o1;
void print(Item &i1);
void display();
void insert();
void search();
void dlt();
bool compare(const Item &i1, const Item &i2)
{
 //if (i1.name != i2.name) return i1.cost < i2.cost;
 return i1.cost < i2.cost;
}
int main()
{
 int ch;
 do
 {
 cout<<"\n* * * * * Menu * * * * *";
 cout<<"\n1.Insert";
 cout<<"\n2.Display";
 cout<<"\n3.Search";
 cout<<"\n4.Sort";
 cout<<"\n5.Delete";
 cout<<"\n6.Exit";
 cout<<"\nEnter your choice : ";
 cin>>ch;

 switch(ch)
 {
 case 1:
 insert();
```

```
 break;

 case 2:
 display();
 break;

 case 3:
 search();
 break;

 case 4:
 sort(o1.begin(),o1.end(),compare);
 cout<<"\n\n Sorted on Cost : ";
 display();
 break;

 case 5:
 dlt();
 break;

 case 6:
 exit(0);
 }

 }while(ch!=7);
 return 0;
}
void insert()
{
 Item i1;
 cout<<"\nEnter Item Name : ";
```

```
 cin>>i1.name;
 cout<<"\nEnter Item Quantity : ";
 cin>>i1.quantity;
 cout<<"\nEnter Item Cost : ";
 cin>>i1.cost;
 cout<<"\nEnter Item Code : ";
 cin>>i1.code;
 o1.push_back(i1);
}
void display()
{
 for_each(o1.begin(),o1.end(),print);
}
void print(Item &i1)
{
 cout<<"\n";
 cout<<"\nItem Name : "<<i1.name;
 cout<<"\nItem Quantity : "<<i1.quantity;
 cout<<"\nItem Cost : "<<i1.cost;
 cout<<"\nItem Code : "<<i1.code;
 cout<<"\n\n";
}
void search()
{
 vector<Item>::iterator p;
 Item i1;
 cout<<"\nEnter Item Code to search : ";
 cin>>i1.code;
 p=find(o1.begin(),o1.end(),i1);
 if(p==o1.end())
 {
```

```
 cout<<"\nNot found!!!";
 }
 else
 {
 cout<<"\nFound!!!";
 }
}
void dlt()
{
 vector<Item>::iterator p;
 Item i1;
 cout<<"\nEnter Item Code to delete : ";
 cin>>i1.code;
 p=find(o1.begin(),o1.end(),i1);
 if(p==o1.end())
 {
 cout<<"\nNot found!!!";
 }
 else
 {
 o1.erase(p);
 cout<<"\nDeleted!!!";
 }
}
```

Program 7

```
#include <iostream>
#include <map>
#include <string>
#include <utility>
```

```
using namespace std;

int main()
{
	typedef map<string,int> mapType;
	mapType populationMap;

	populationMap.insert(pair<string, float>("Maharashtra", 125));
	populationMap.insert(pair<string, float>("Uttar Pradesh", 225));
	populationMap.insert(mapType::value_type("Bihar", 120));
	populationMap.insert(mapType::value_type("West Bengal", 100));
	populationMap.insert(make_pair("Madhya Pradesh", 90));
	populationMap.insert(make_pair("Tamil Nadu", 80));
	populationMap.insert(make_pair("Rajasthan", 78));
	populationMap.insert(make_pair("Andhra Pradesh", 53));
	populationMap.insert(make_pair("Odisha", 47));
	populationMap.insert(make_pair("Kerala", 38));
	populationMap.insert(make_pair("Telangana", 37));
	populationMap.insert(make_pair("Assam", 35));
	populationMap.insert(make_pair("Jharkhand", 38));
	populationMap.insert(make_pair("Karnataka", 68));
	populationMap.insert(make_pair("Gujarat", 70));
	populationMap.insert(make_pair("Punjab", 31));
	populationMap.insert(make_pair("Chhattisgarh", 30));
	populationMap.insert(make_pair("Haryana", 29));
	populationMap.insert(make_pair("UT Delhi", 19));
	populationMap.insert(make_pair("UT Jammu and Kashmir", 14));
	populationMap.insert(make_pair("Uttarakhand", 12));
	populationMap.insert(make_pair("Himachal Pradesh", 8));
	populationMap.insert(make_pair("Tripura", 04));
	populationMap.insert(make_pair("Meghalaya", 4));
```

```
        populationMap.insert(make_pair("Manipur[", 3));
        populationMap.insert(make_pair("Nagaland", 2));
        populationMap.insert(make_pair("Goa", 2));
        populationMap.insert(make_pair("Arunachal Pradesh", 2));
        populationMap.insert(make_pair("UT Puducherry", 2));
        populationMap.insert(make_pair("Mizoram", 1));
        populationMap.insert(make_pair("UT Chandigarh", 1));
        populationMap.insert(make_pair("Sikkim", 1));
        populationMap.insert(make_pair("UT Dadra and Nagar Haveli and Daman and Diu", 1));
        populationMap.insert(make_pair("UT Andaman and Nicobar Islands", 1));
        populationMap.insert(make_pair("UT Lakshadweep", 0.0003));
        populationMap.insert(make_pair("UT Ladakh", 0.00006));

        mapType::iterator iter = --populationMap.end();
        populationMap.erase(iter);

        cout << "Total state and UT of India with Size of populationMap: " << populationMap.size()
<< '\n';

        for (iter = populationMap.begin(); iter != populationMap.end(); ++iter)
        {
         cout << iter->first <<":" << iter->second << " million\n";
        }

        char c;
        do
        {
                string state;
                cout<<"\nEnter that state you want to know the population of: ";
                cin>>state;
                iter = populationMap.find(state);
```

```
            if( iter != populationMap.end() )
               cout << state <<"'s populations is "
                  << iter->second << " million\n";
            else
               cout << "State is not in populationMap" << '\n';

            cout<<"Do you wish to continue?(y/n):";
            cin>>c;
        }while(c=='y'||c=='Y');

        populationMap.clear();

        return 0;
}
```

Experiment No. 1

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

```
#include <conio.h>
#include <iostream>
#include <graphics.h>
#include <stdlib.h>
using namespace std;
class point
{
 public:
 int x,y;
};
class poly
{
 private:
 point p[20];
 int inter[20],x,y;
 int v,xmin,ymin,xmax,ymax;
 public:
 int c;
 void read();
 void calcs();
 void display();
 void ints(float);
 void sort(int);
};
void poly::read()
{
 int i;
 cout<<"\n Scan Fill Algorithm ";
 cout<<"\n Enter Number Of Vertices Of Polygon: ";
 cin>>v;
```

```
 if(v>2)
 {
 for(i=0;i<v; i++) //ACCEPT THE VERTICES
 {
 cout<<"\nEnter co-ordinate no. "<<i+1<<" : ";
 cout<<"\n\tx"<<(i+1)<<"=";
 cin>>p[i].x;
 cout<<"\n\ty"<<(i+1)<<"=";
 cin>>p[i].y;
 }
 p[i].x=p[0].x;
 p[i].y=p[0].y;
 xmin=xmax=p[0].x;
 ymin=ymax=p[0].y;
 }
 else
 cout<<"\n Enter valid no. of vertices.";
}
void poly::calcs()
{
 for(int i=0;i<v;i++)
 {
 if(xmin>p[i].x)
 xmin=p[i].x;
 if(xmax<p[i].x)
 xmax=p[i].x;
 if(ymin>p[i].y)
 ymin=p[i].y;
 if(ymax<p[i].y)
 ymax=p[i].y;
 }
}
void poly::display()
{
 int ch1;
 char ch='y';
```

```
 float s,s2;
 do
 {
 cout<<"\n\nMENU:";
 cout<<"\n\n\t1 . Scan line Fill ";
 cout<<"\n\n\t2 . Exit ";
 cout<<"\n\nEnter your choice:";
 cin>>ch1;
 switch(ch1)
 {
 case 1:
 s=ymin+0.01;
 delay(100);
 cleardevice();
 while(s<=ymax)
 {
 ints(s);
 sort(s);
 s++;
 }
 break;
 case 2:
 exit(0);
 }
 cout<<"Do you want to continue?: ";
 cin>>ch;
 }while(ch=='y' || ch=='Y');
}
void poly::ints(float z)
{
 int x1,x2,y1,y2,temp;
 c=0;
 for(int i=0;i<v;i++)
 {
 x1=p[i].x;
 y1=p[i].y;
```

```
x2=p[i+1].x;
y2=p[i+1].y;
if(y2<y1)
{
temp=x1;
x1=x2;
x2=temp;
temp=y1;
y1=y2;
y2=temp;
}
if(z<=y2&&z>=y1)
{
if((y1-y2)==0)
x=x1;
else
{
x=((x2-x1)*(z-y1))/(y2-y1);
x=x+x1;
}
if(x<=xmax && x>=xmin)
inter[c++]=x;
}
}
}
void poly::sort(int z) // sorting
{
int temp,j,i;
for(i=0;i<v;i++)
{
line(p[i].x,p[i].y,p[i+1].x,p[i+1].y);
}
delay(100);
for(i=0; i<c;i+=2)
{
delay(100);
```

```
 line(inter[i],z,inter[i+1],z);
 }
}
int main() //main
{
 int cl;
 initwindow(500,600);
 cleardevice();
 poly x;
 x.read();
 x.calcs();
 cleardevice();
 cout<<"\n\tEnter The Color You Want :(In Range 0 To 15 )->"; //selecting color
 cin>>cl;
 setcolor(cl);
 x.display();

 closegraph(); //closing graph
 getch();
 return 0;
}
```

Input :

Number of Vertices : 4

Cordinates 1st :
x1= 200
y1= 200

Cordinates 2st :
x2= 200
y2= 400

Cordinates 3st :
x3= 400

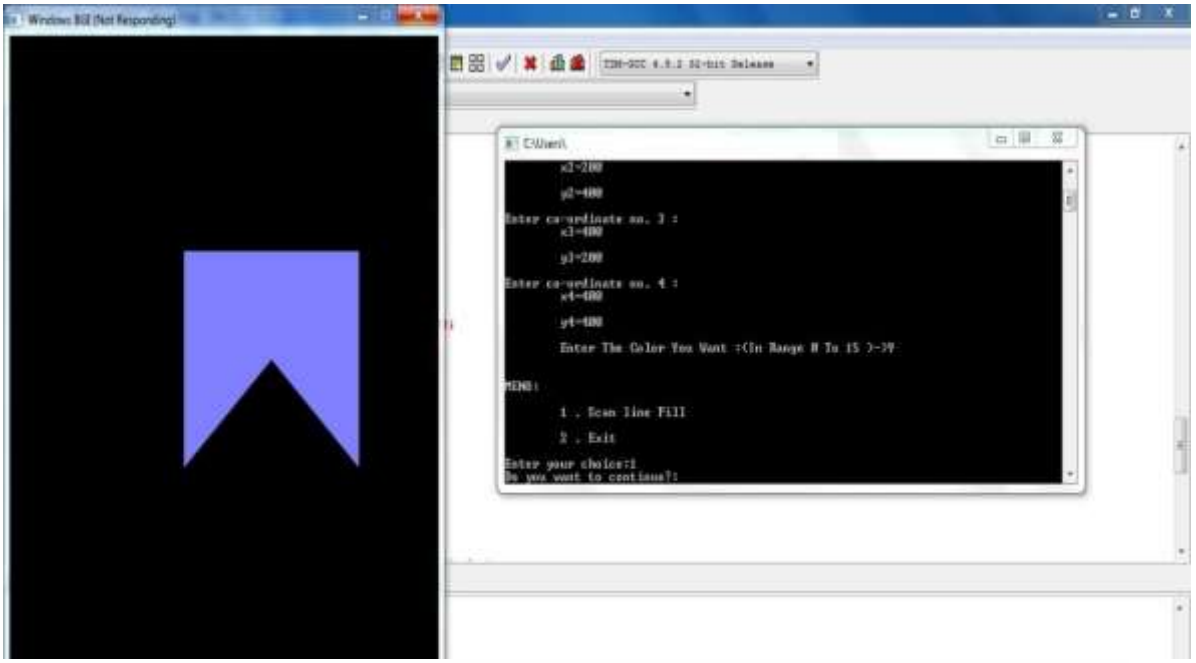
y3= 200

Cordinates 4st :

x4= 400

y4= 400

Output :



Experiment No. 2

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

Code :

```
#include<iostream>
#include<stdlib.h>
#include<math.h>
#include<graphics.h>
#include<dos.h>
using namespace std;
class Coordinate
{
	public:
		int x,y;
		char code[4];
};
class Lineclip
{
	public:
		Coordinate PT;
		void drawwindow();
		void drawline(Coordinate p1,Coordinate p2);
		Coordinate setcode(Coordinate p);
		int visibility(Coordinate p1,Coordinate p2);
		Coordinate resetendpt(Coordinate p1,Coordinate p2);
};
int main()
{
	Lineclip lc;
	int gd = DETECT,v,gm;
	Coordinate p1,p2,p3,p4,ptemp;
	cout<<"\n Enter x1 and y1\n";
	cin>>p1.x>>p1.y;
	cout<<"\n Enter x2 and y2\n";
	cin>>p2.x>>p2.y;
```

```
        initgraph(&gd,&gm,"");
        lc.drawwindow();
        delay(2000);
        lc.drawline (p1,p2);
        delay(2000);
        cleardevice();
        delay(2000);
        p1=lc.setcode(p1);
        p2=lc.setcode(p2);
        v=lc.visibility(p1,p2);
        delay(2000);
        switch(v)
        {
                case 0: lc.drawwindow();
                                delay(2000);
                                lc.drawline(p1,p2);
                                break;
           case 1:lc.drawwindow();
                delay(2000);
                break;
           case 2:p3=lc.resetendpt(p1,p2);
                  p4=lc.resetendpt(p2,p1);
                  lc.drawwindow();
                  delay(2000);
                  lc.drawline(p3,p4);
                  break;
  }
   delay(2000);
   closegraph();
}

void Lineclip::drawwindow()
{
        line(150,100,450,100);
        line(450,100,450,350);
```

```
        line(450,350,150,350);
        line(150,350,150,100);
 }

 void Lineclip::drawline(Coordinate p1,Coordinate p2)
 {
        line(p1.x,p1.y,p2.x,p2.y);
 }
Coordinate Lineclip::setcode(Coordinate p)
{
        Coordinate ptemp;
        if(p.y<100)
           ptemp.code[0]='1';
        else
           ptemp.code[0]='0';

        if(p.y>350)
                ptemp.code[1]='1';
        else
                ptemp.code[1]='0';
        if(p.x>450)
                ptemp.code[2]='1';
        else
                ptemp.code[2]='0';

        if(p.x<150)
                ptemp.code[3]='1';
        else
                ptemp.code[3]='0';
                ptemp.x=p.x;
        ptemp.y=p.y;
        return(ptemp);
};
int Lineclip:: visibility(Coordinate p1,Coordinate p2)
{
        int i,flag=0;
```

```
	for(i=0;i<4;i++)
	{
		if(p1.code[i]!='0' || (p2.code[i]=='1'))
		  flag='0';
	}
	if(flag==0)
	 return(0);

		for(i=0;i<4;i++)
	{
		if(p1.code[i]==p2.code[i] && (p2.code[i]=='1'))
		 flag='0';
	}

	if(flag==0)
		return(1);
		return(2);
}
Coordinate Lineclip::resetendpt(Coordinate p1,Coordinate p2)
{
	Coordinate temp;
	int x,y,i;
	float m,k;
		if(p1.code[3]=='1')
		x=150;
	if(p1.code[2]=='1')
		x=450;
	if((p1.code[3]=='1') || (p1.code[2])=='1')
	{
		m=(float)(p2.y-p1.y)/(p2.x-p1.x);
		k=(p1.y+(m*(x-p1.x)));
		temp.y=k;
		temp.x=x;
		for(i=0;i<4;i++)
		temp.code[i]=p1.code[i];
```

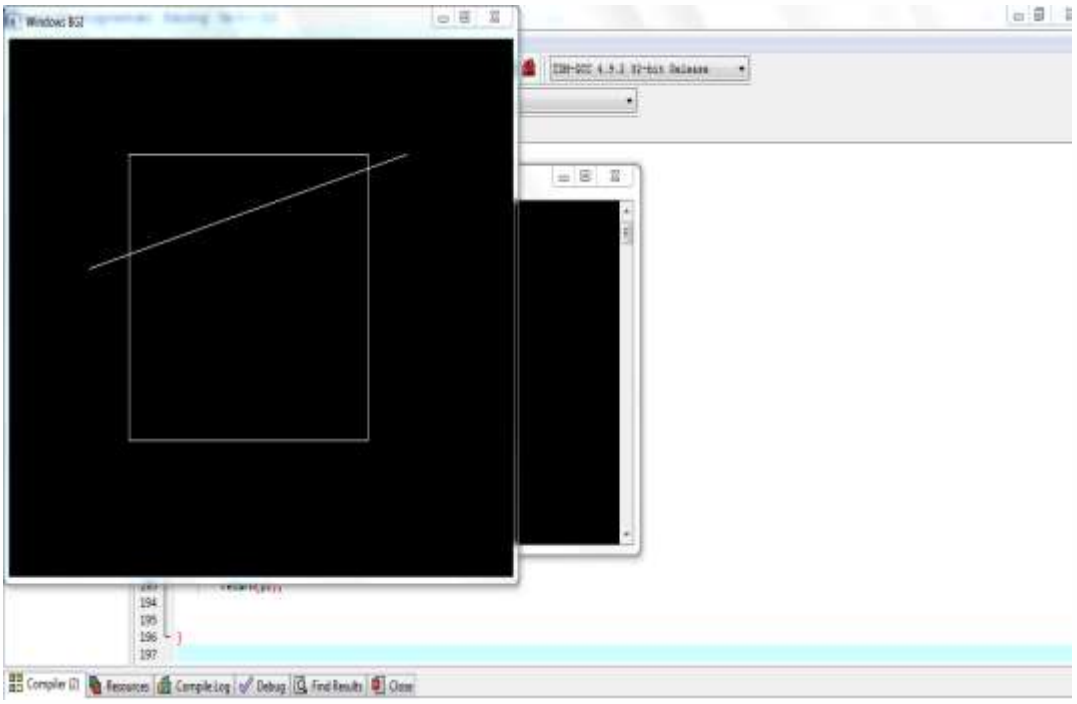
```
    if(temp.y<=350 && temp.y>=100)
        return (temp);
}
if(p1.code[0]=='1')
        y=100;
if(p1.code[1]=='1')
        y=350;
if((p1.code[1]=='1') || (p1.code[1]=='1'))
{
        m=(float)(p2.y-p1.y)/(p2.x-p1.x);
        k=(float)p1.x+(float)(y-p1.y)/m;
        temp.x=k;
        temp.y=y;

        for(i=0;i<4;i++)
                temp.code[i]=p1.code[i];

        return(temp);
}
else
        return(p1);
}
```

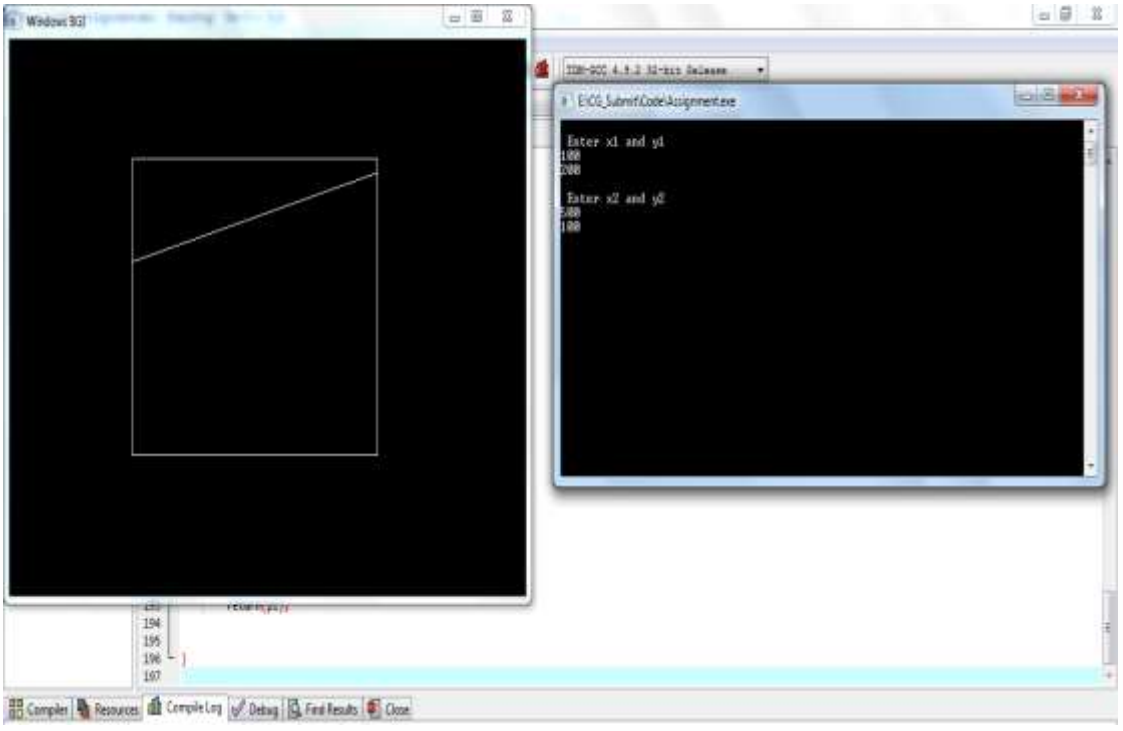
Input :

X1 , Y1:
100
200

X2, Y2 :
500
100

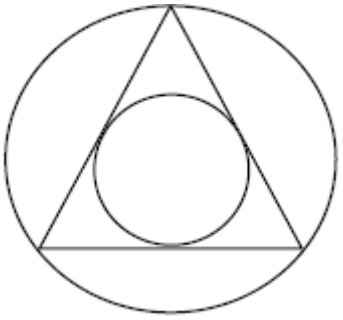
Output :




E:\CG_Submit\Code\Assignment.exe
Enter x1 and y1
100
200
Enter x2 and y2
500
100

Experiment No. 3

a) **Write C++ program to draw the following pattern. Use DDA line and Bresenham's circle drawing algorithm. Apply the concept of encapsulation.**



Code :

```
#include <iostream>
# include <graphics.h>
# include <stdlib.h>
using namespace std;
class dcircle
{
private: int x0, y0;
public:
dcircle()
{
x0=0;
y0=0;
}
void setoff(int xx, int yy)
{
x0=xx;
y0=yy;
}
void drawc(int x1, int y1, int r)
{
float d;
int x,y;
```

```
x=0;
y=r;
d=3-2*r;
do
{
putpixel(x1+x0+x, y0+y-y1, 15);
putpixel(x1+x0+y, y0+x-y1,15);
putpixel(x1+x0+y, y0-x-y1,15);
putpixel(x1+x0+x,y0-y-y1,15);
putpixel(x1+x0-x,y0-y-y1,15);
putpixel(x1+x0-y, y0-x-y1,15);
putpixel(x1+x0-y, y0+x-y1,15);
putpixel(x1+x0-x, y0+y-y1,15);
if (d<=0)
{
d = d+4*x+6;
}
else
{
d=d+4*(x-y)+10;
y=y-1;
}
x=x+1;
}
while(x<y);
}
};

class pt
{
protected: int xco, yco,color;
public:
pt()
{
xco=0,yco=0,color=15;
}
```

```
void setco(int x, int y)
{
xco=x;
yco=y;
}
void setcolor(int c)
{
color=c;
}
void draw()
{
putpixel(xco,yco,color);
}
};
class dline:public pt
{
private: int x2, y2;
public:
dline():pt()
{
x2=0;
y2=0;
}
void setline(int x, int y, int xx, int yy)
{
pt::setco(x,y);
x2=xx;
y2=yy;
}
void drawl( int colour)
{
float x,y,dx,dy,length;
int i;
pt::setcolor(colour);
dx= abs(x2-xco);
dy=abs(y2-yco);
```

```
if(dx>=dy)
{
length= dx;
}
else
{
length= dy;
}
dx=(x2-xco)/length;
dy=(y2-yco)/length;
x=xco+0.5;
y=yco+0.5;
i=1;
while(i<=length)
{
pt::setco(x,y);
pt::draw();
x=x+dx;
y=y+dy;
i=i+1;
}
pt::setco(x,y);
pt::draw();
}
};
int main()
{
int gd=DETECT, gm;
initgraph(&gd, &gm, NULL);
int x,y,r, x1, x2, y1, y2, xmax, ymax, xmid, ymid, n, i;
dcircle c;
cout<<"\nenter coordinates of centre of circle : ";
cout<<"\n enter the value of x : ";
cin>>x;
cout<<"\nenter the value of y : ";
cin>>y;
```

```
cout<<"\nenter the value of radius : ";
cin>>r;
xmax= getmaxx();
ymax=getmaxy();
xmid=xmax/2;
ymid=ymax/2;
setcolor(1);
c.setoff(xmid,ymid);
line(xmid, 0, xmid, ymax);
line(0,ymid,xmax,ymid);
setcolor(15);
c.drawc(x,y,r);
pt p1;
p1.setco(100,100);
p1.setcolor(14);
dline l;
l.setline(x1+xmid, ymid-y1, x2+xmid, ymid-y2);
cout<<"Enter Total Number of lines : ";
cin>>n;
for(i=0;i<n;i++)
{
cout<<"Enter co-ordinates of point x1 : ";
cin>>x1;
cout<<"enter coordinates of point y1 : ";
cin>>y1;
cout<<"Enter co-ordinates of point x2 : ";
cin>>x2;
cout<<"enter coordinates of point y2 : ";
cin>>y2;
l.setline(x1+xmid, ymid-y1, x2+xmid, ymid-y2);
l.drawl(15);
}
cout<<"\nEnter coordinates of centre of circle : ";
cout<<"\n Enter the value of x : ";
cin>>x;
cout<<"\nEnter the value of y : ";
```

```
cin>>y;
cout<<"\nEnter the value of radius : ";
cin>>r;
setcolor(5);
c.drawc(x,y,r);
getch();
delay(200);
closegraph();
return 0;
}
```

Input :

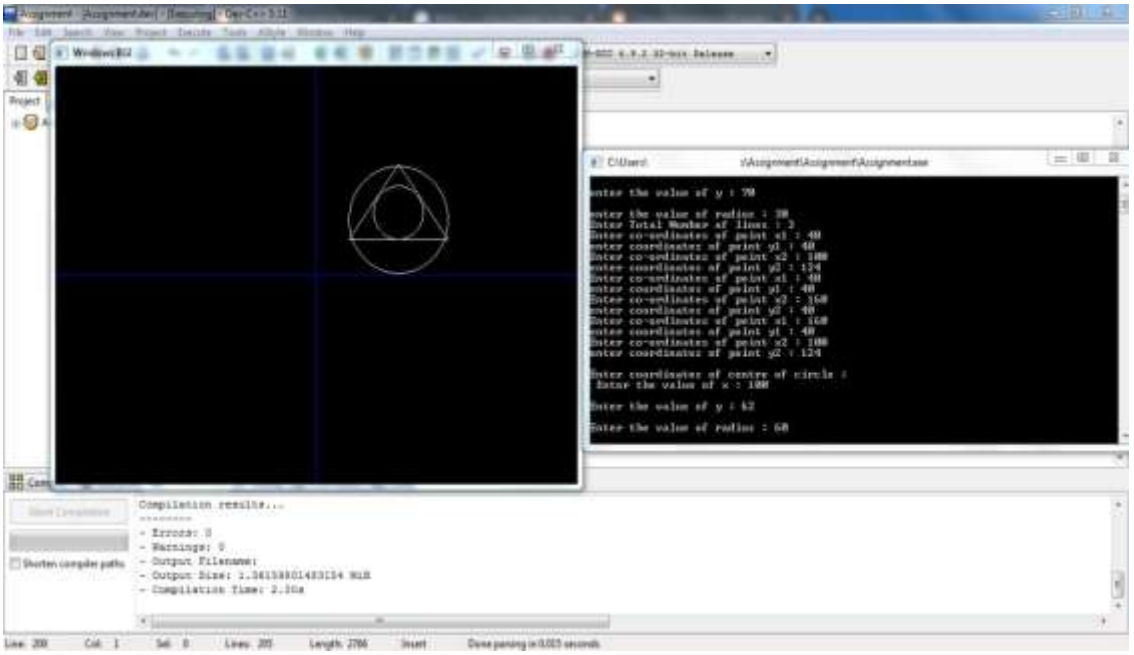
Value Of X : 100

Value Of Y : 70

Value Of R : 30

Next Inputs In Image Given Below.

Output :



Experiment No. 4

Write C++ program to draw 2-D object and perform following basic transformation

a) Scaling

b) Translation

c) Rotation

Apply the concept of operator overloading.

Code :

```
#include<iostream>
#include<graphics.h>
#include<math.h>
using namespace std;
class transform
{
	public:
		int m,a[20][20],c[20][20];
		int i,j,k;
		public:
		void object();
		void accept();
		void operator *(float b[20][20])
		{
			for(int i=0;i<m;i++)
			{
				for(int j=0;j<m;j++)
				{
					c[i][j]=0;
					for(int k=0;k<m;k++)
					{
						c[i][j]=c[i][j]+(a[i][k]*b[k][j]);
```

```
                                        }
                                    }
                                }
                            }
};
void transform::object()
{
          int gd,gm;
          gd=DETECT;
          initgraph(&gd,&gm,NULL);
  line(300,0,300,600);
  line(0,300,600,300);
   for( i=0;i<m-1;i++)
   {
          line(300+a[i][0],300-a[i][1],300+a[i+1][0],300-a[i+1][1]);
          }
          line(300+a[0][0],300-a[0][1],300+a[i][0],300-a[i][1]);
          for( i=0;i<m-1;i++)
          {
                    line(300+c[i][0],300-c[i][1],300+c[i+1][0],300-c[i+1][1]);
          }
          line(300+c[0][0],300-c[0][1],300+c[i][0],300-c[i][1]);
          int temp;
          cout << "Press 1 to continue";
          cin >> temp;
          closegraph();
}
void transform::accept()
{
cout<<"\n";
 cout<<"Enter the Number Of Edges:";
    cin>>m;
    cout<<"\nEnter The Coordinates :";
    for(int i=0;i<m;i++)
    {
          for(int j=0;j<3;j++)
```

```
        {
                if(j>=2)
                a[i][j]=1;
                else
                cin>>a[i][j];
                }
        }
}
int main()
{
        int ch,tx,ty,sx,sy;
        float deg,theta,b[20][20];
        transform t;
        t.accept();

          cout<<"\nEnter your choice";
          cout<<"\n1.Translation"
              "\n2.Scaling"
                      "\n3.Rotation";
                      cin>>ch;
                switch(ch)
                {
                case 1: cout<<"\nTRANSLATION OPERATION\n";
                      cout<<"Enter value for tx and ty:";
                      cin>>tx>>ty;
                      b[0][0]=b[2][2]=b[1][1]=1;
                              b[0][1]=b[0][2]=b[1][0]=b[1][2]=0;
                              b[2][0]=tx;
                              b[2][1]=ty;
                              t * b;
                              t.object();
                              break;
          case 2: cout<<"\nSCALING OPERATION\n";
                      cout<<"Enter value for sx,sy:";
                      cin>>sx>>sy;
                      b[0][0]=sx;
```

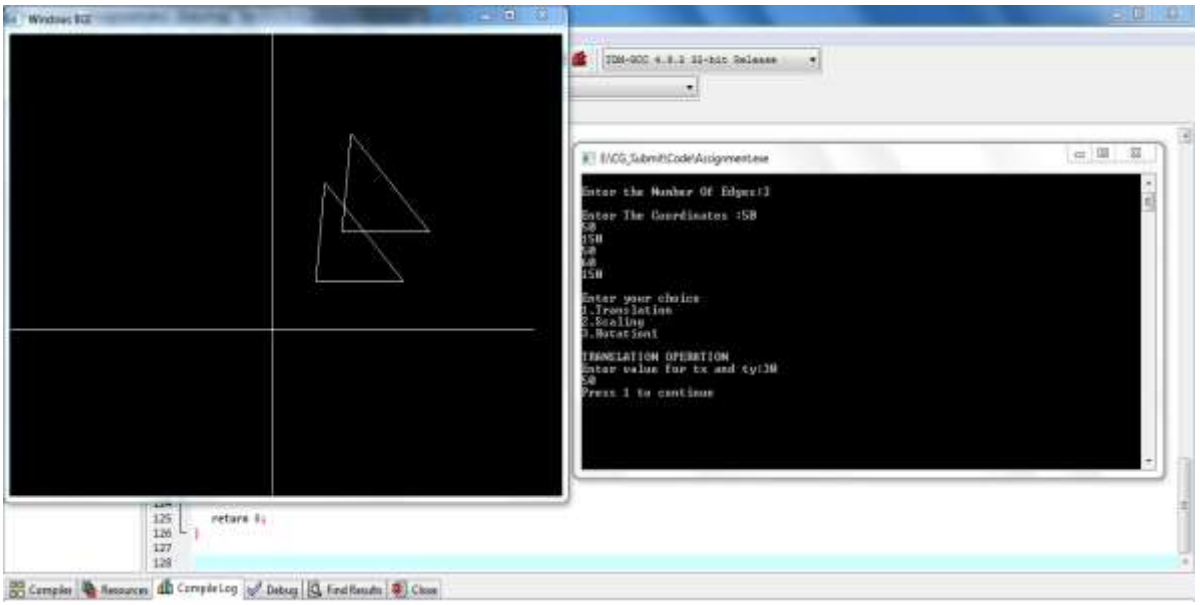
```
                b[1][1]=sy;
                b[0][1]=b[0][2]=b[1][0]=b[1][2]=0;
                b[2][0]=b[2][1]=0;
                        b[2][2] = 1;
                        t * b;
                        t.object();
                        break;
            case 3: cout<<"\nROTATION OPERATION\n";
                cout<<"Enter value for angle:";
                cin>>deg;
                        theta=deg*(3.14/100);
                        b[0][0]=b[1][1]=cos(theta);
                        b[0][1]=sin(theta);
                        b[1][0]=sin(-theta);
                        b[0][2]=b[1][2]=b[2][0]=b[2][1]=0;
                        b[2][2]=1;
                        t * b;
                        t.object();
                        break;
            default:
               cout<<"\nInvalid choice";
            }
        getch();
    return 0;
}
```

Input :

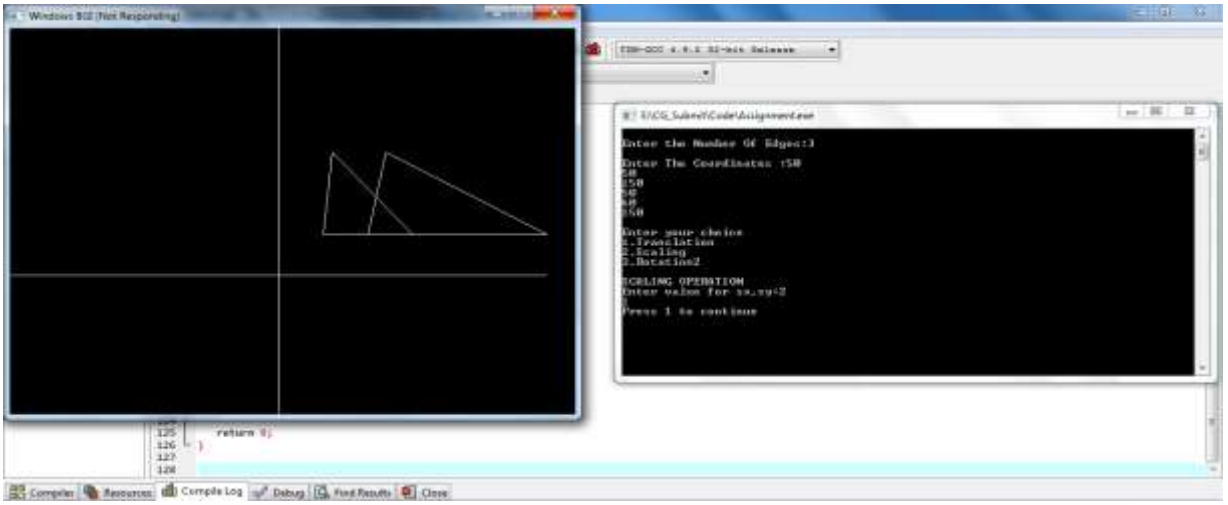
Provided In Image Given Below

Output :

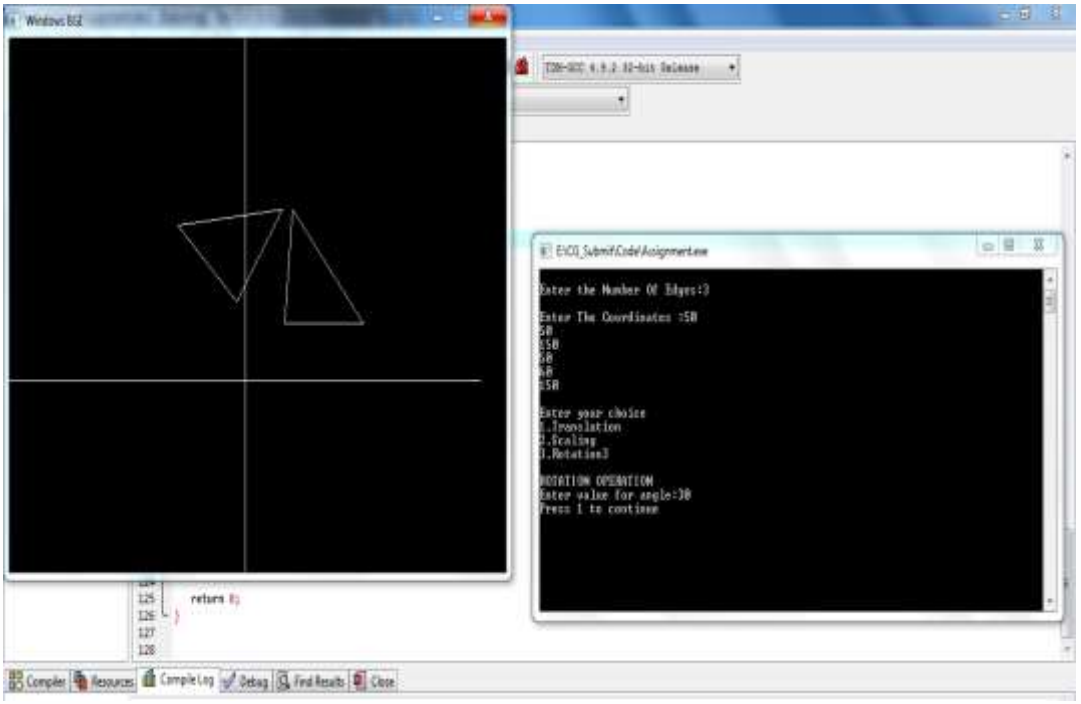
For Tranlation :


Enter the Number Of Edges:3
Enter The Coordinates :50
50
150
250
50
150
Enter your choice
1.Translation
2.Scaling
3.Rotation1
TRANSLATION OPERATION
Enter value for tx and ty:30
50
Press 1 to continue

For Scaling :


Enter the Number Of Edges:3
Enter The Coordinates :50
50
150
250
50
150
Enter your choice
1.Translation
2.Scaling
3.Rotation2
SCALING OPERATION
Enter value for sx,sy:2
2
Press 1 to continue

For Rotation :



Experiment No. 5

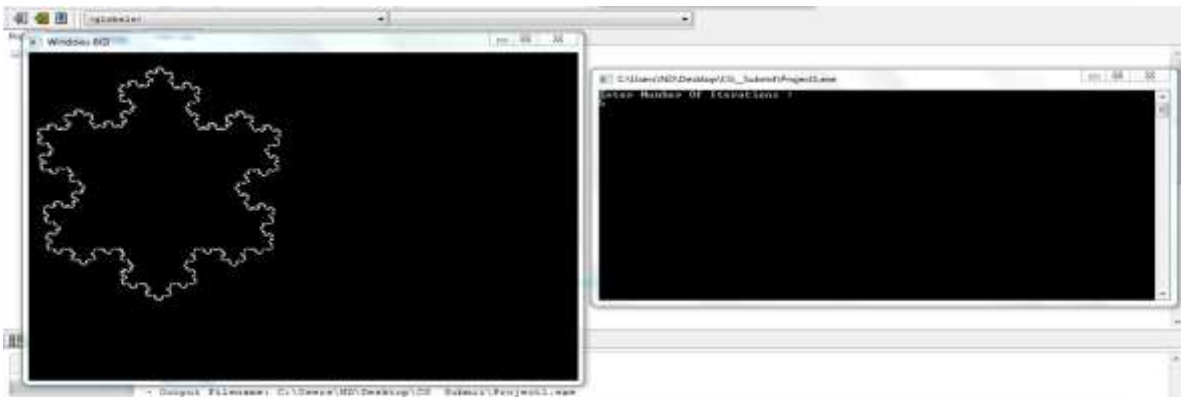
Write C++ Program To Generate Fractal Patterns By Using Koch Curves

Code :

```
#include <iostream>
#include <math.h>
#include <graphics.h>
using namespace std;
class kochCurve
{
public:
void koch(int it,int x1,int y1,int x5,int y5)
{
int x2,y2,x3,y3,x4,y4;
int dx,dy;
if (it==0)
{
line(x1,y1,x5,y5);
}
else
{
delay(10);
dx=(x5-x1)/3;
dy=(y5-y1)/3;
x2=x1+dx;
y2=y1+dy;
x3=(int)(0.5*(x1+x5)+sqrt(3)*(y1-y5)/6);
y3=(int)(0.5*(y1+y5)+sqrt(3)*(x5-x1)/6);
x4=2*dx+x1;
y4=2*dy+y1;
koch(it-1,x1,y1,x2,y2);
```

```
koch(it-1,x2,y2,x3,y3);
koch(it-1,x3,y3,x4,y4);
koch(it-1,x4,y4,x5,y5);
}
}
};
int main()
{
kochCurve k;
int it;
cout<<"Enter Number Of Iterations : "<<endl;
cin>>it;
int gd=DETECT,gm;
initgraph(&gd,&gm,NULL);
k.koch(it,150,20,20,280);
k.koch(it,280,280,150,20);
k.koch(it,20,280,280,280);
getch();
closegraph();
return 0;
}
```

Output :



Experiment No. 6

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

Code :

```
#include<iostream>
#include<math.h>
#include<GL/glut.h>
using namespace std;
typedef float Matrix4 [4][4];
Matrix4 theMatrix;
static GLfloat input[8][3]=
{
 {40,40,-50},{90,40,-50},{90,90,-50},{40,90,-50},
 {30,30,0},{80,30,0},{80,80,0},{30,80,0}
};
float output[8][3];
float tx,ty,tz;
float sx,sy,sz;
float angle;
int choice,choiceRot;
void setIdentityM(Matrix4 m)
{
for(int i=0;i<4;i++)
 for(int j=0;j<4;j++)
 m[i][j]=(i==j);
}
void translate(int tx,int ty,int tz)
{
for(int i=0;i<8;i++)
{
```

```
output[i][0]=input[i][0]+tx;
output[i][1]=input[i][1]+ty;
output[i][2]=input[i][2]+tz;
}
}
void scale(int sx,int sy,int sz)
{
 theMatrix[0][0]=sx;
 theMatrix[1][1]=sy;
 theMatrix[2][2]=sz;
}
void RotateX(float angle) //Parallel to x
{
angle = angle*3.142/180;
 theMatrix[1][1] = cos(angle);
theMatrix[1][2] = -sin(angle);
theMatrix[2][1] = sin(angle);
theMatrix[2][2] = cos(angle);
}
void RotateY(float angle) //parallel to y
{
angle = angle*3.14/180;
theMatrix[0][0] = cos(angle);
theMatrix[0][2] = -sin(angle);
theMatrix[2][0] = sin(angle);
theMatrix[2][2] = cos(angle);
}
void RotateZ(float angle) //parallel to z
{
angle = angle*3.14/180;
theMatrix[0][0] = cos(angle);
theMatrix[0][1] = sin(angle);
theMatrix[1][0] = -sin(angle);
theMatrix[1][1] = cos(angle);
}
void multiplyM()
```

```
{
//We Don't require 4th row and column in scaling and rotation
//[8][3]=[8][3]*[3][3] //4th not used
for(int i=0;i<8;i++)
{
 for(int j=0;j<3;j++)
 {
 output[i][j]=0;
 for(int k=0;k<3;k++)
 {
 output[i][j]=output[i][j]+input[i][k]*theMatrix[k][j];
 }
 }
}
}
void Axes(void)
{
glColor3f (0.0, 0.0, 0.0); // Set the color to BLACK
glBegin(GL_LINES); // Plotting X-Axis
glVertex2s(-1000 ,0);
glVertex2s( 1000 ,0);
glEnd();
glBegin(GL_LINES); // Plotting Y-Axis
glVertex2s(0 ,-1000);
glVertex2s(0 , 1000);
glEnd();
}
void draw(float a[8][3])
{
 glBegin(GL_QUADS);
 glColor3f(0.7,0.4,0.5); //behind
 glVertex3fv(a[0]);
 glVertex3fv(a[1]);
 glVertex3fv(a[2]);
 glVertex3fv(a[3]);
 glColor3f(0.8,0.2,0.4); //bottom
```

```
 glVertex3fv(a[0]);
 glVertex3fv(a[1]);
 glVertex3fv(a[5]);
 glVertex3fv(a[4]);
 glColor3f(0.3,0.6,0.7); //left
 glVertex3fv(a[0]);
 glVertex3fv(a[4]);
 glVertex3fv(a[7]);
 glVertex3fv(a[3]);
 glColor3f(0.2,0.8,0.2); //right
glVertex3fv(a[1]);
glVertex3fv(a[2]);
glVertex3fv(a[6]);
glVertex3fv(a[5]);
glColor3f(0.7,0.7,0.2); //up
glVertex3fv(a[2]);
glVertex3fv(a[3]);
glVertex3fv(a[7]);
glVertex3fv(a[6]);
glColor3f(1.0,0.1,0.1);
glVertex3fv(a[4]);
glVertex3fv(a[5]);
glVertex3fv(a[6]);
glVertex3fv(a[7]);
glEnd();
}
void init()
{
 glClearColor(1.0,1.0,1.0,1.0); //set backgrond color to white
 glOrtho(-454.0,454.0,-250.0,250.0,-250.0,250.0);
 // Set the no. of Co-ordinates along X & Y axes and their gappings
 glEnable(GL_DEPTH_TEST);
 // To Render the surfaces Properly according to their depths
}
void display()
{
```

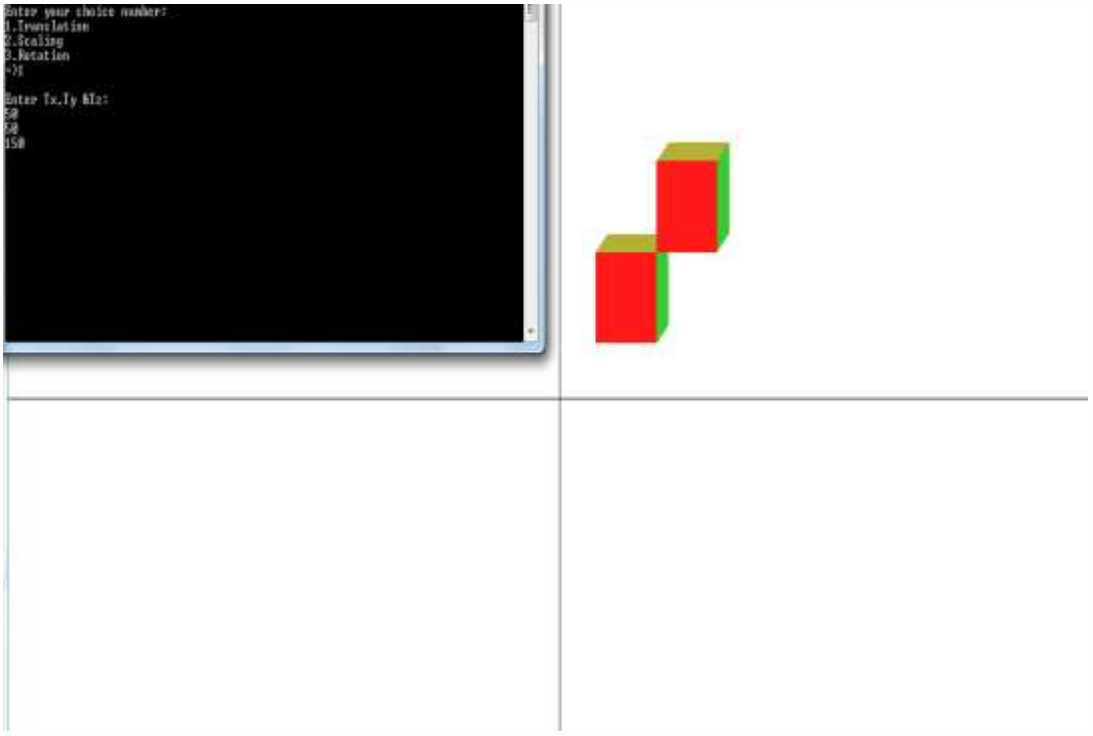
```
glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
Axes();
glColor3f(1.0,0.0,0.0);
draw(input);
setIdentityM(theMatrix);
switch(choice)
{
case 1:
 translate(tx,ty,tz);
 break;
case 2:
 scale(sx,sy,sz);
multiplyM();
 break;
case 3:
 switch (choiceRot) {
 case 1:
 RotateX(angle);
 break;
 case 2: RotateY(angle);
 break;
 case 3:
 RotateZ(angle);
 break;
 default:
 break;
 }
multiplyM();
 break;
}
draw(output);
glFlush();
}
int main(int argc, char** argv)
{
 glutInit(&argc,argv);
```

```
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB|GLUT_DEPTH);
glutInitWindowSize(1362,750);
glutInitWindowPosition(0,0);
glutCreateWindow("3D TRANSFORMATIONS");
init();
cout<<"Enter your choice number:\n1.Translation\n2.Scaling\n3.Rotation\n=>";
cin>>choice;
switch (choice) {
case 1:
cout<<"\nEnter Tx,Ty &Tz: \n";
cin>>tx>>ty>>tz;
break;
case 2:
cout<<"\nEnter Sx,Sy & Sz: \n";
cin>>sx>>sy>>sz;
break;
case 3:
cout<<"Enter your choice for Rotation about axis:\n1.parallel to X-axis."
<<"(y& z)\n2.parallel to Y-axis.(x& z)\n3.parallel to Z-axis."
<<"(x& y)\n =>";
cin>>choiceRot;
switch (choiceRot) {
case 1:
cout<<"\nENter Rotation angle: ";
cin>>angle;
break;
case 2:
cout<<"\nENter Rotation angle: ";
cin>>angle;
break;
case 3:
cout<<"\nENter Rotation angle: ";
cin>>angle;
break;
default:
break;
```

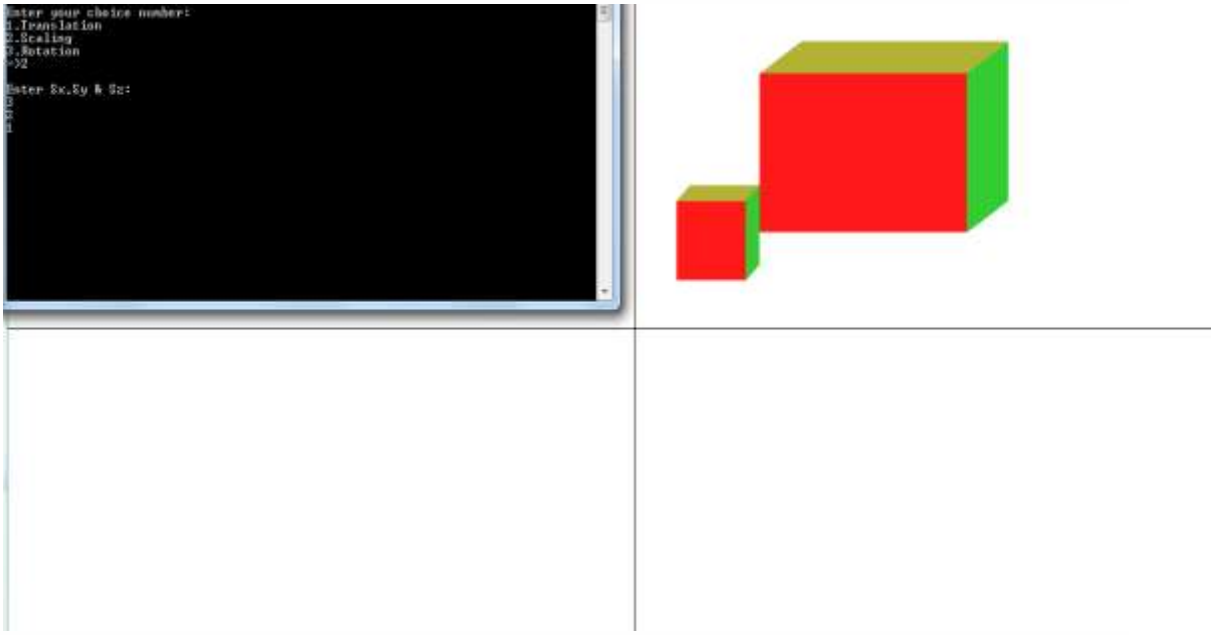
```
 }
 break;
 default:
 break;
 }
 glutDisplayFunc(display);
 glutMainLoop();
return 0;
}
```

Output :

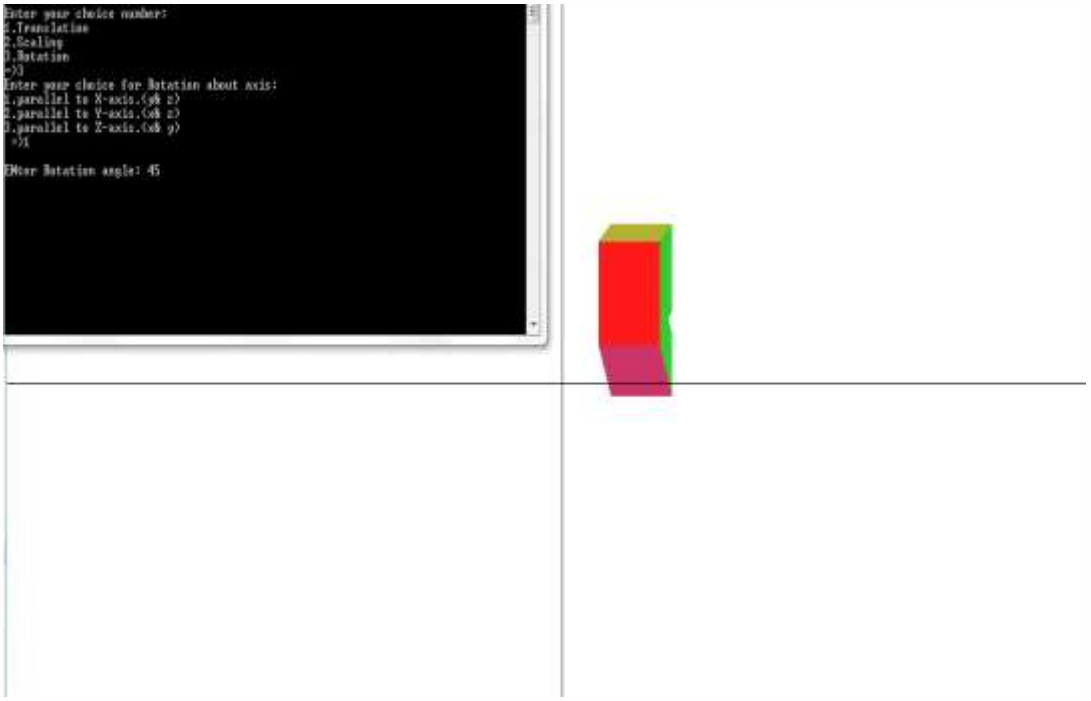
1. Translation :



2. Scaling :



3. Rotation:



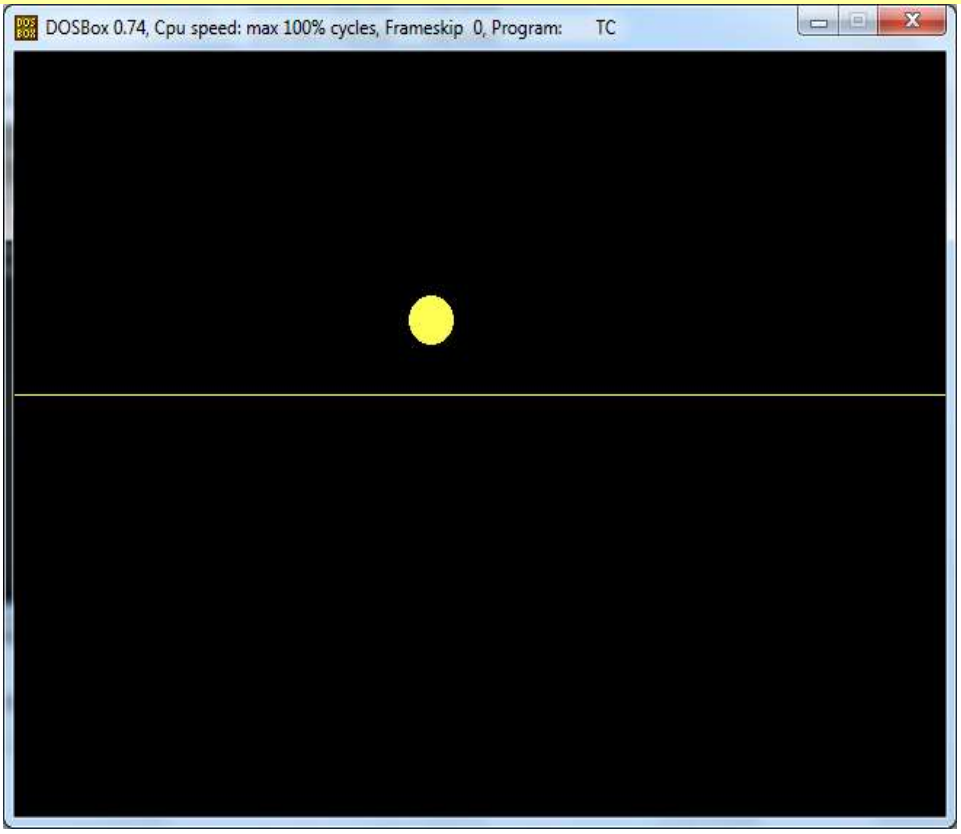
Experiment No. 7

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

```
#include<dos.h>
#include<iostream.h>
#include<graphics.h>
#include<math.h>
#include<conio.h>
void main()
 {
   int d=DETECT,m;
   initgraph(&d,&m,"e:\tcc\bgi");
   float x=1,y=0.00000,j=.5,count=.1;
   float r=15;
   setcolor(14);
   line(0,215,650,215);
    sleep(1);
    for(int k=0;k<=7;k++)
     {

      for(float i=90;i<270;i+=10)
       {
 y=cos(((i*22/7)/180))/j;
```

```
if(y>0)
y=-y;
x+=5;
setcolor(14);
setfillstyle(1,14);
circle(x,y*100+200,r);
floodfill(x,y*100+200,14);
  delay(100);
setcolor(0);
setfillstyle(1,0);
circle(x,y*100+200,r);
floodfill(x,y*100+200,0);
     }
     j+=count;
     count+=.1;
    }
  getch();
}
```

Output



```
#include<conio.h>
#include"graphics.h"
#include"stdlib.h"
void main()
{
  intgd=DETECT,gm,i=0,x,xx,y,yy,r;
  //Initializes the graphics system
  initgraph(&gd,&gm,"c:\\tc\\bgi");
  x=getmaxx();
  y=getmaxy();
  while(!kbhit())
  {
    i++;
   //   setfillstyle(random(i),random(30));

    circle(xx=random(x),yy=random(y),random(30));
    setfillstyle(random(i),random(30));
    floodfill(xx,yy,getmaxcolor());
    delay(200);
  }
  getch();
}
```