/*

Title: - Write C++ program to draw a concave polygon and fill it with desired color using fill algorithm.

Class:-SE Computer

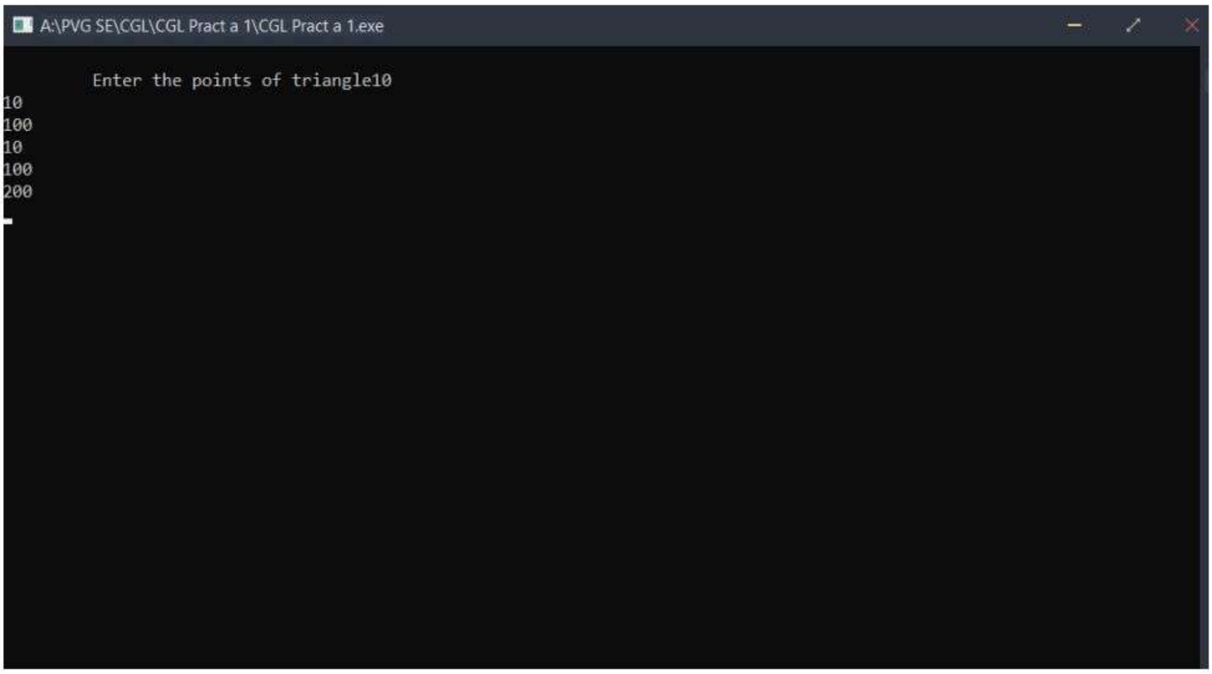
Sub:-OOPL & CGL

***/

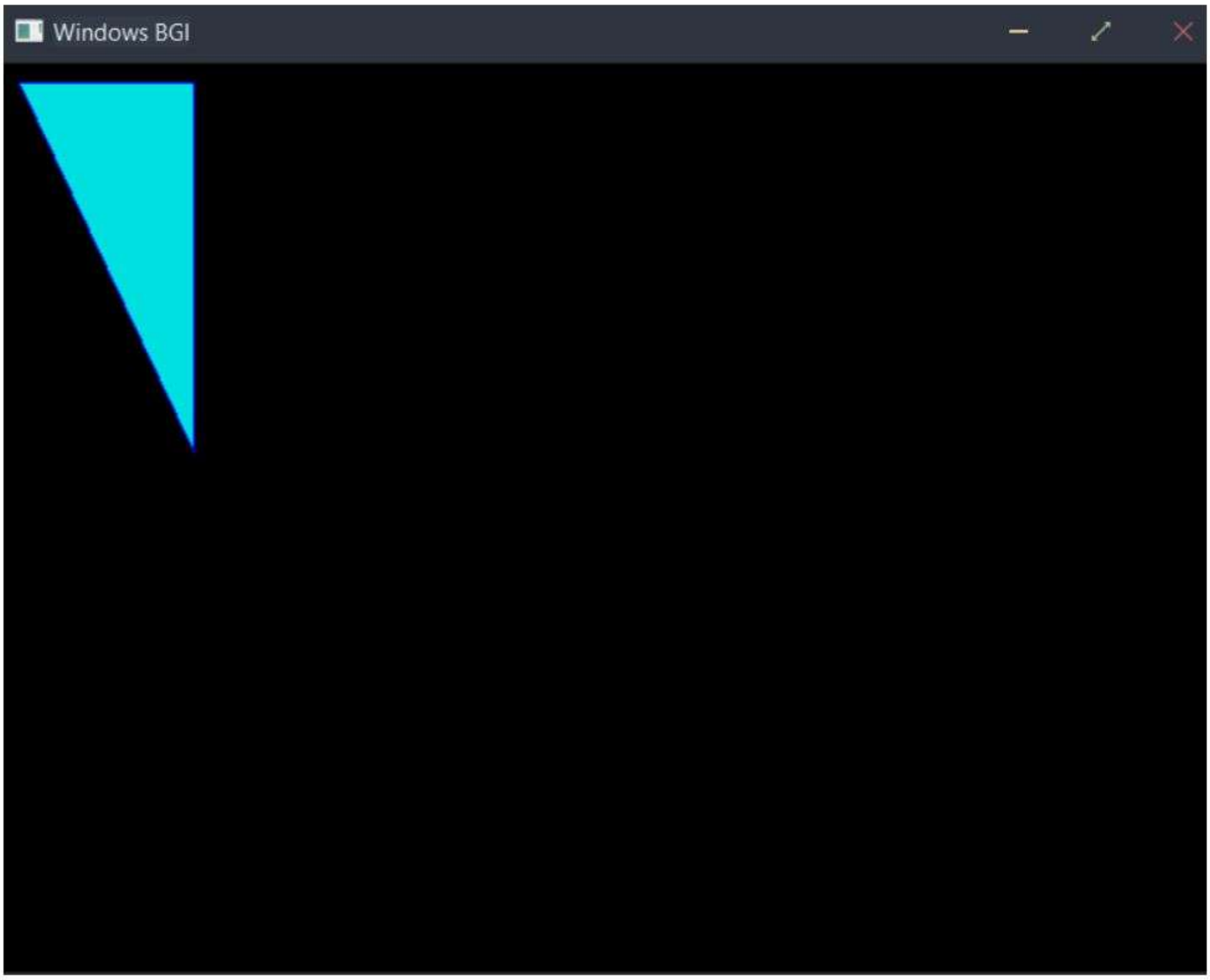
Program-

```
#include<graphics.h>
#include<iostream>
#include<stdlib.h>
using namespace std;
void ffill(int x,int y,int o_col,int n_col)
{
int current = getpixel(x,y);
if(current==o_col)
{
delay(1);
putpixel(x,y,n_col);
ffill(x+1,y,o_col,n_col);
ffill(x-1,y,o_col,n_col);
ffill(x,y+1,o_col,n_col);
ffill(x,y-1,o_col,n_col);
}
}
int main()
{
int x1,y1,x2,y2,x3,y3,xavg,yavg;
int gdriver = DETECT,gmode;
initgraph(&gdriver,&gmode,NULL);
cout << " \n\t Enter the points of triangle";
setcolor(1);
cin >> x1 >> y1 >> x2 >> y2 >> x3 >> y3;
xavg = (int)(x1+x2+x3)/3;
yavg = (int)(y1+y2+y3)/3;
line(x1,y1,x2,y2);
line(x2,y2,x3,y3);
line(x3,y3,x1,y1);
ffill(xavg,yavg,0,3);
getch();
return 0;
}
```

/*Output:-



```
        Enter the points of triangle10
10
100
10
100
200
```


Windows BGI

/*

Title: - Write C++ program to generate Hilbert curve using concept of fractals.

Roll No:-

Class:-SE Computer

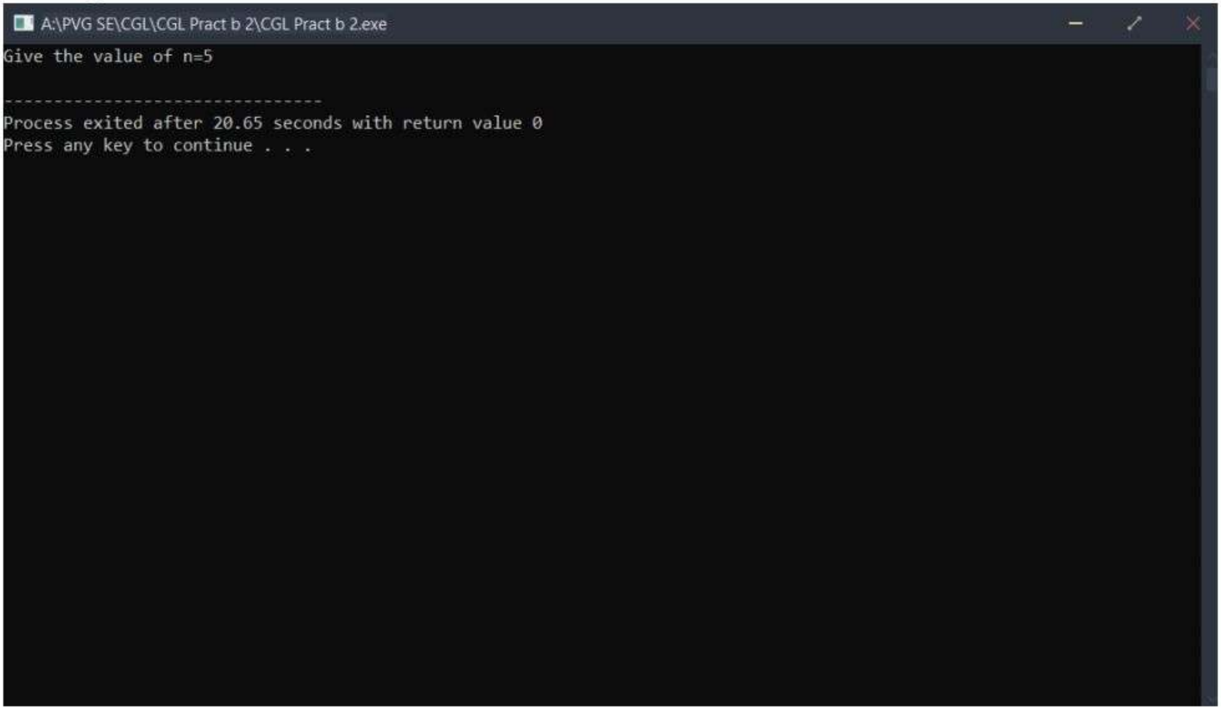
Sub:-OOPL & CGL

**/

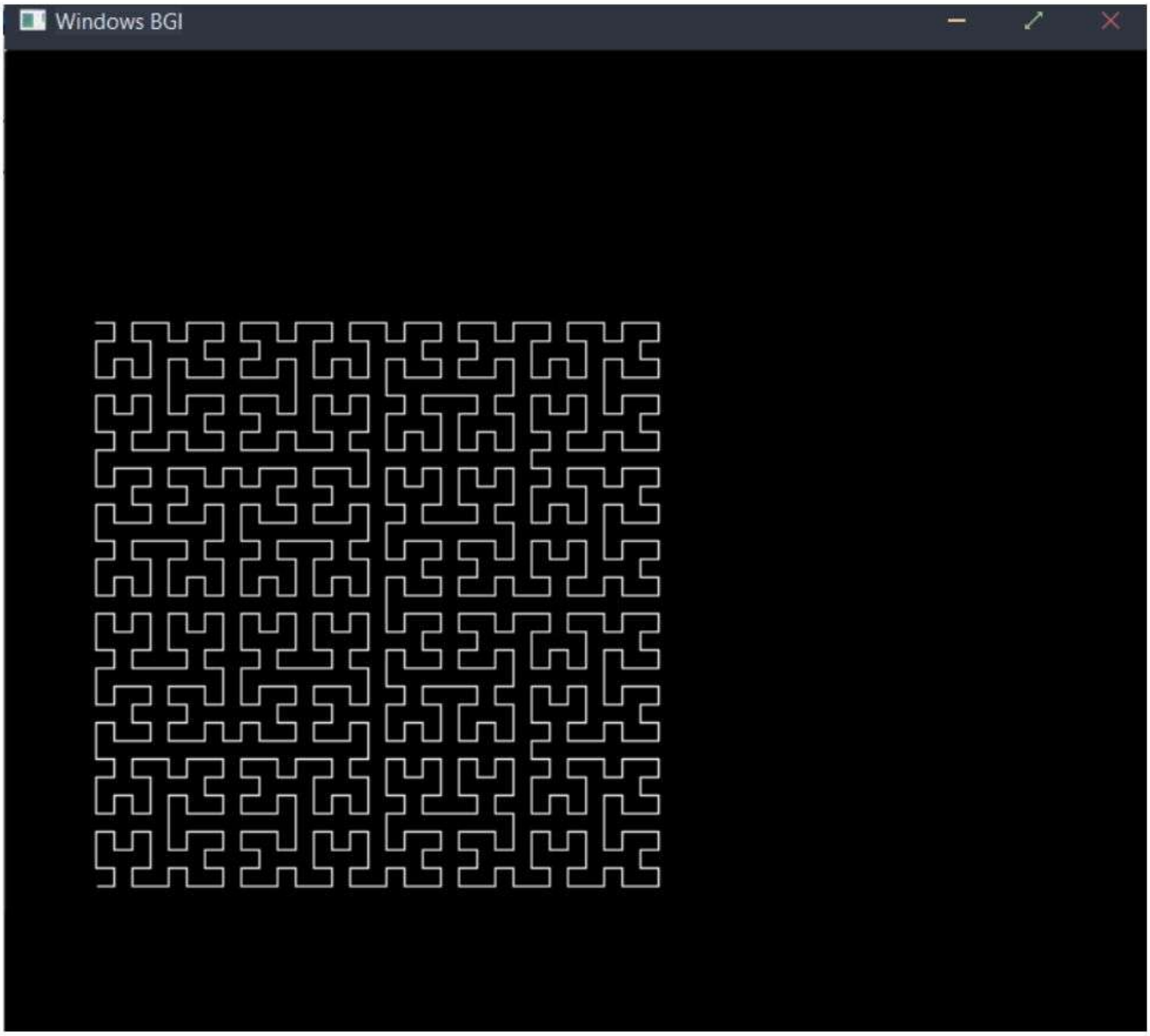
Program-

```
#include<iostream>
#include<graphics.h>
#include<math.h>
#include<cstdlib>
using namespace std;
void move(int j, int h, int &x,int &y)
{
        if(j==1)
        y-=h;
        else
                if(j==2)
                        x+=h;
                else if(j==3)
                        y+=h;
                else if(j==4)
                        x-=h;
        lineto(x,y);
}
void hilbert(int r,int d,int l ,int u,int i,int h,int &x,int &y)
{
        if(i>0)
        {
                i--;
                hilbert(d,r,u,l,i,h,x,y);
                move(r,h,x,y);
                hilbert(r,d,l,u,i,h,x,y);
                move(d,h,x,y);
                hilbert(r,d,l,u,i,h,x,y);
                move(l,h,x,y);
                hilbert(u,l,d,r,i,h,x,y);
        }
}
int main()
{
        int n,x1,y1;
        int x0=50,y0=150,x,y,h=10,r=2,d=3,l=4,u=1;
        cout<<"Give the value of n=";
        cin>>n;
        x=x0;
        y=y0;
        int driver=DETECT,mode=0;
        initgraph(&driver,&mode,NULL);
        moveto(x,y);
        hilbert(r,d,l,u,n,h,x,y);
        delay(10000);
        closegraph();
        return 0;
}
```

/*Output:-



```
Give the value of n=5

--------------------------------
Process exited after 20.65 seconds with return value 0
Press any key to continue . . .
```


Windows BGI

/*

Title: - Write OpenGL Program to draw Sunrise and Sun Set.

Roll No:-

Class:-SE Computer

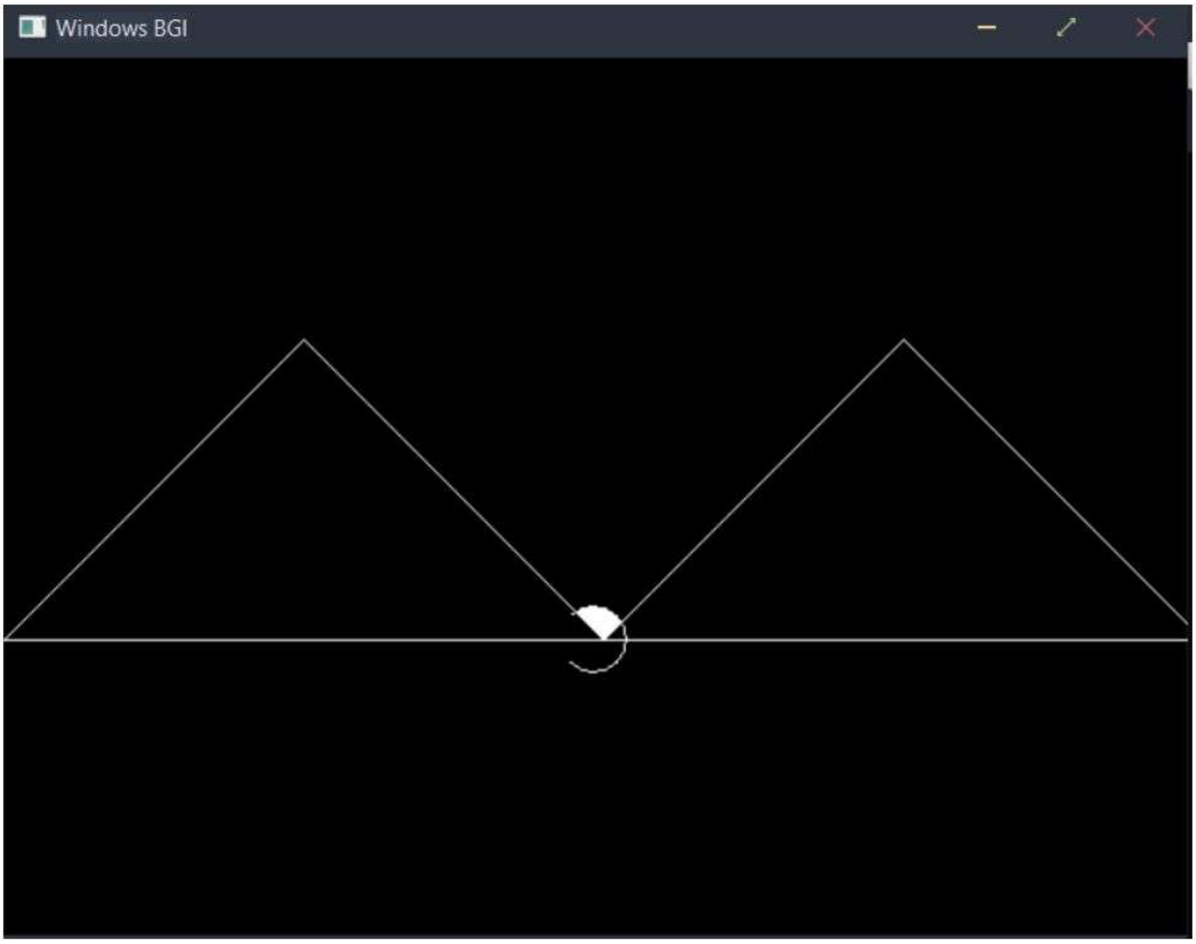
Sub:-OOPL & CGL

**/

Program-

```
#include<graphics.h>
int main()
{
        int gd = DETECT, gm;
        initgraph(&gd, &gm,NULL);
        int midx,midy,r=10;
        midx=getmaxx()/2;
        while(r<=50)
        {
                cleardevice();
                setcolor(WHITE);
                line(0,310,160,150);
                line(160,150,320,310);
                line(320,310,480,150);
                line(480,150,640,310);
                line(0,310,640,310);
                arc(midx,310,225,133,r);
                floodfill(midx,300,15);
                if(r>20)
                {
                        setcolor(7);
                        floodfill(2,2,15);
                        setcolor(6);
                        floodfill(150,250,15);
                        floodfill(550,250,15);
                        setcolor(2);
                        floodfill(2,450,15);
                }
                delay(1000);
                r+=2;
        }
        getch();
        closegraph();
}
```

/*Output:-



/*

Mini Project

Title: - Write a C++ program to draw a man walking in rain with an umbrella.

Roll No:-

Class:-SE Computer

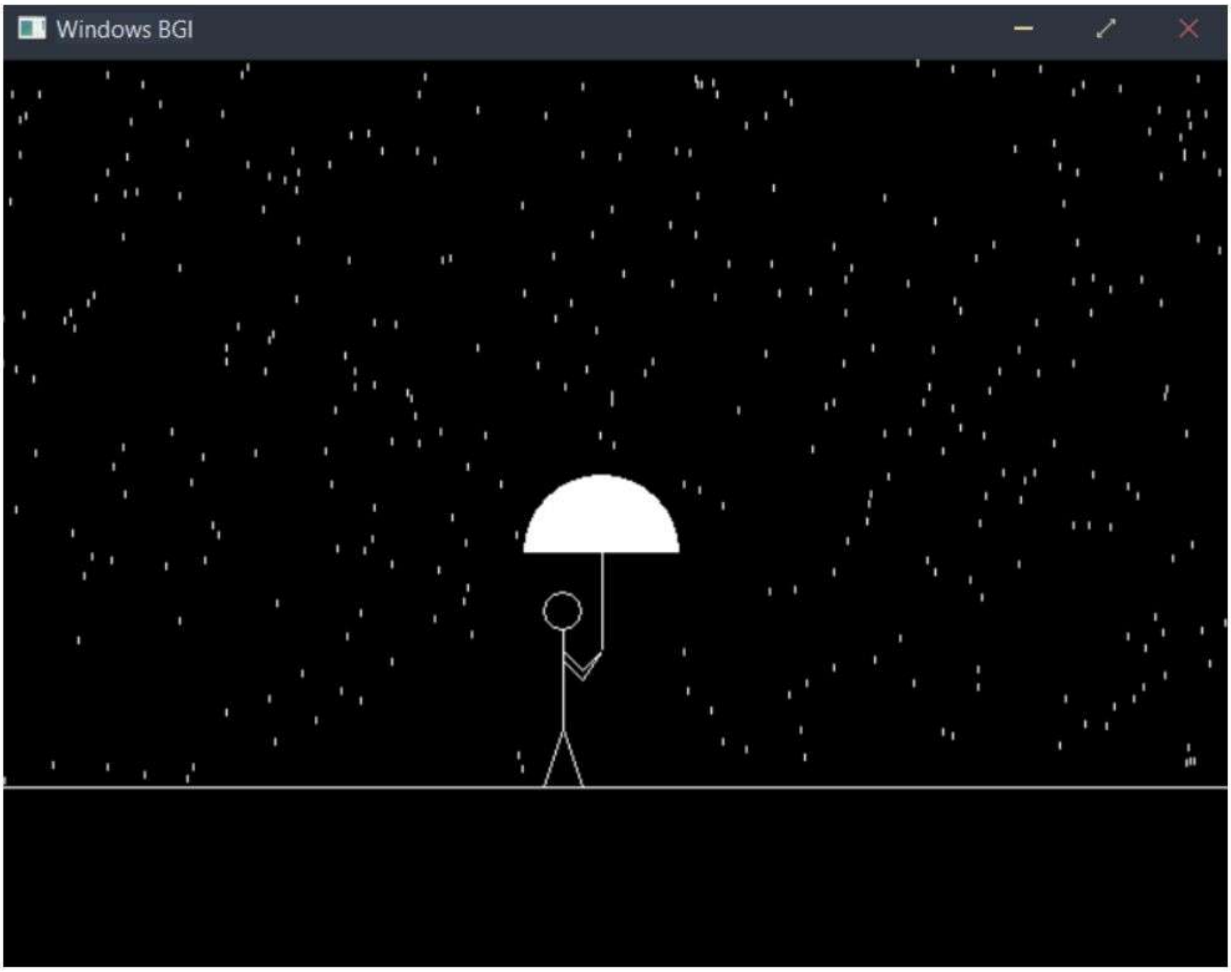
Sub:-OOPL & CGL

**/

Program-

```
#include<stdio.h>
#include<graphics.h>
#define ScreenWidth getmaxx()
#define ScreenHeight getmaxy()
#define GroundY ScreenHeight*0.80
int ldisp=0;
void DrawManAndUmbrella(int x,int ldisp)
{
//Man's head
circle(x,GroundY-90,10);
line(x,GroundY-80,x,GroundY-30);
//Man's hand
line(x,GroundY-70,x+10,GroundY-60);
line(x,GroundY-65,x+10,GroundY-55);
line(x+10,GroundY-60,x+20,GroundY-70);
line(x+10,GroundY-55,x+20,GroundY-70);
//Man's legs
line(x,GroundY-30,x+ldisp,GroundY);
line(x,GroundY-30,x-ldisp,GroundY);
//umbrella
pieslice(x+20,GroundY-120,0,180,40);
line(x+20,GroundY-120,x+20,GroundY-70);
}
void Rain(int x)
{
int i,rx,ry;
for(i=0;i<400;i++)
{
        rx=rand() % ScreenWidth;
        ry=rand() % ScreenHeight;
        if(ry<GroundY-4)
        {
                if(ry<GroundY-120 || (ry>GroundY-120 && (rx<x-20 || rx>x+60)))
                line(rx,ry,rx+0.5,ry+4);
        }
}
}
int main()
{
int gd=DETECT,gm,x=0;
initgraph(&gd,&gm,NULL);
   while(!kbhit())
   {
        //Draw Ground
        line(0,GroundY,ScreenWidth,GroundY);
        Rain(x);
   ldisp=(ldisp+2)%20;
        DrawManAndUmbrella(x,ldisp);
   delay(75);
   cleardevice();
   x=(x+2)%ScreenWidth;
}
   getch();
}
```

/*Output:-



```
#include<iostream>

#include<dos.h>

#include<stdlib.h>

#include<math.h>

#include<graphics.h>

/* Defining structure for end point of line */
using namespace std;
typedef struct coordinate
{

        int x;

        int y;

        char code[4];

}PT;

void drawwindow();

void drawline (PT p1,PT p2,int cl);
```

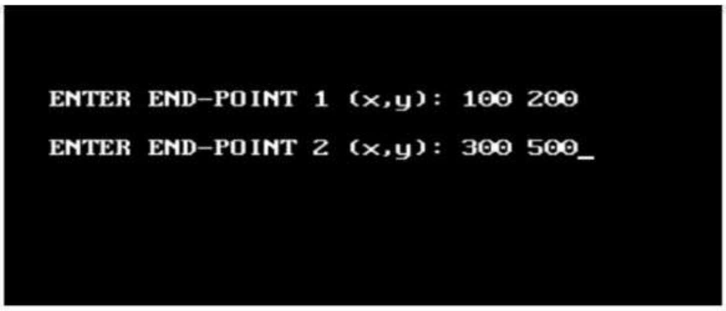
```
PT setcode(PT p);

int visibility (PT p1,PT p2);

PT resetendpt (PT p1,PT p2);

void check_line(PT p1,PT p2);

int main()

{
        initwindow(800,800);
        //int gd=DETECT, gm;

        PT p1,p2;

        cout<<"\n\t\tENTER END-POINT 1 (x,y): ";

        cin>>p1.x>>p1.y;

        cout<<"\n\t\tENTER END-POINT 2 (x,y): ";

        cin>>p2.x>>p2.y;

        //initgraph(&gd,&gm,"\\Turboc3\\bgi");

        drawwindow();

        drawline(p1,p2,15);
        check_line(p1,p2);

        return(0);

}

void check_line(PT p1,PT p2)

{

        int v;



        p1=setcode(p1);

        p2=setcode(p2);

        v=visibility(p1,p2);

        switch(v)

        {

                case 0: cleardevice(); /* Line conpletely visible */

                drawwindow();

                drawline(p1,p2,15);
        break;

        case 1: cleardevice(); /* Line completely invisible */

        drawwindow();

        break;

        case 2: cleardevice(); /* line partly visible */

        p1=resetendpt (p1,p2);

        p2=resetendpt(p2,p1);

        check_line(p1,p2);

        break;

    }

    delay(2000);

}

/* Function to draw window */

void drawwindow()
```

```
{
        setcolor(RED);
        line(150,100,450,100);
        line(450,100,450,350);
        line(450,350,150,350);
        line(150,350,150,100);
        delay(2000);
}
/* Function to draw line between two points
-------------------------------------------*/
void drawline (PT p1,PT p2,int cl)
{
        setcolor(cl);

        line(p1.x,p1.y,p2.x,p2.y);
        delay(2000);

}/* Function to set code of the coordinates

---------------------------------------------*/

PT setcode(PT p)

{

        PT ptemp;

        if(p.y<100)

        ptemp.code[0]='1'; /* TOP */

        else

        ptemp.code[0]='0';

        if(p.y>350)

        ptemp.code[1]='1'; /* BOTTOM */

        else

        ptemp.code[1]='0';

        if (p.x>450)
        ptemp.code[2]='1'; /* RIGHT */

        else

        ptemp.code[2]='0';

        if (p.x<150) /* LEFT */

        ptemp.code[3]='1';

        else

        ptemp.code[3]='0';

        ptemp.x=p.x;

        ptemp.y=p.y;

        return(ptemp);

}

/* Function to determine visibility of line

--------------------------------------------*/

int visibility (PT p1,PT p2)
```

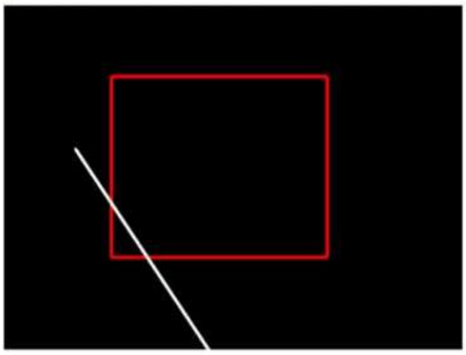
```
{
    int i,flag=0;
    for(i=0;i<4;i++)
    {
        if((p1.code[i]!='0')||(p2.code[i]!='0'))

        flag=2;
    }
    for(i=0;i<4;i++)
    {
        if((p1.code[i]==p2.code[i]) &&(p1.code[i]=='1'))
        flag=1;
    }
    if(flag==0)
        return(0);
        if(flag==1)

                return(1);

        if(flag==2)

                return(2);

}

/* Function to find new end points

---------------------------------------*/

PT resetendpt (PT p1,PT p2)

{

        PT temp;

        int x,y,i;

        float m,k;

        if( p1.code[3]=='1') /* Cutting LEFT Edge */

                x=150;

        if(p1.code[2]=='1') /* Cutting RIGHT Edge */
```

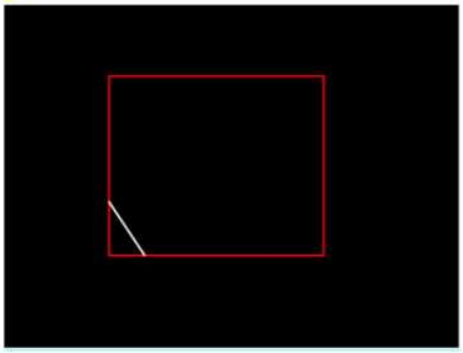
```
            x=450;

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;

            if(temp.y<=350&&temp.y>=100)

                        return(temp);

}

if(p1.code[0]=='1') /* Cutting TOP Edge */

            y=100;

if(p1.code [1]=='1') /* Cutting BOTTOM Edge */

            y=350;
```

```
    if((p1.code[0]=='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;
        if(temp.x<=450&&temp.x>=150)
            return(temp);
    }
    else
    return(p1);
}
```



```
ENTER END-POINT 1 (x,y): 100 200
ENTER END-POINT 2 (x,y): 300 500_
```





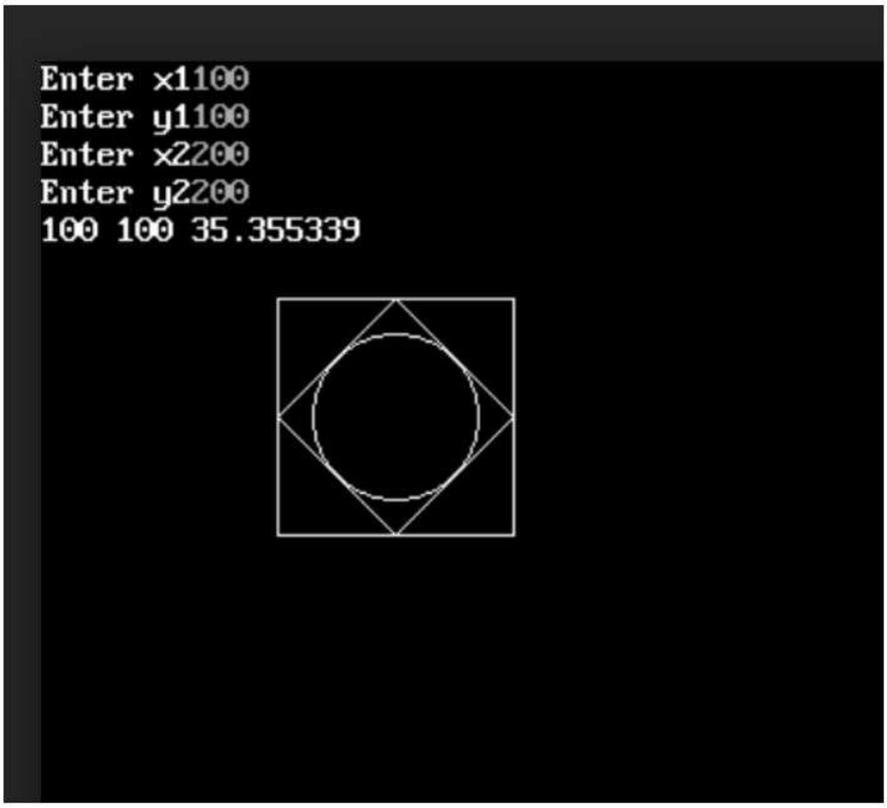
```
#include<conio.h>
#include<iostream>
#include<graphics.h>
#include<math.h>
using namespace std;
class drawpattern
{
	private:
	float dx,dy,i ,length;
	float count;
	public:
	int x1,y1,x2,y2;
	int xmid,ymid;
	void getdata();
	void ddaline(int x1,int x2,int y1, int y2);
	int xc,yc,r;
	void bdrawcircle(int xc,int yc,int r);
};
void drawpattern::getdata()
{
	cout<<"Enter x1";
	cin>>x1;
	cout<<"Enter y1";
	cin>>y1;
	cout<<"Enter x2";
	cin>>x2;
	cout<<"Enter y2";
	cin>>y2;
}
```

```
void drawpattern::ddaline(int x1, int x2, int y1, int y2)
{
        float x,y;
        dx = (x2-x1);
        dy = (y2-y1);
        //cout<<"value of dx:"<<dx<<endl;
        // cout<<"value of dy:"<<dy<<endl;
        if(abs(dx)>=abs(dy)) length = abs(dx);
        else length = abs(dy);
        // cout<<"length:"<<length<<endl;
        dx = dx/length;
        dy = dy/length;
        x=x1;
        y=y1;
        i=1;
        // cout<<"x"<<" "<<"y"<<"\tPlot(x,y)"<<endl;
        //cout<<"\tplot("<<x<<","<<y<<")"<<endl;
        while(i<=length)
        {
                x = x + dx;
                y = y + dy;
                // cout<<x<<" "<<y;
                // cout<<"\tplot("<<(int)x<<","<<(int)y<<")"<<endl;
                putpixel(x,y,15);
                i++;
        }
}
void drawpattern::bdrawcircle(int xc,int yc,int r)
{
```

```
//xc=320;
//yc=240;
int x,y,d;
x=0;
y=r;
putpixel(xc+x,yc-y,15);
// initialize the decision variable
d=3-2*r;
do
{
        putpixel(xc+x,yc+y,15);
        putpixel(xc-x,yc-y,15);
        putpixel(xc+x,yc-y,15);
        putpixel(xc-x,yc+y,15);
        putpixel(xc+y,yc-x,15);
        putpixel(xc-y,yc-x,15);
        putpixel(xc+y,yc+x,15);
        putpixel(xc-y,yc+x,15);
        if(d<0)
        {
                y=y;
                d=d+4*x+6;
        }
        else
        {
                d=d+4*(x-y)+10;
                y=y-1;
        }
        x=x+1;
```

```
    }
    while(x<=y);
}
int main()
{
    //clrscr();
    initwindow(800,800);
    //int gdriver= DETECT, gmode;
    //initgraph(&gdriver,&gmode,"c://Turboc3//BGI");
    cleardevice();
    drawpattern d;
    d.getdata();
    d.ddaline(d.x1,d.y1,d.x2,d.y1);// (x1,y1) and (x2,y1)
    d.ddaline(d.x2,d.y1,d.x2,d.y2);
    d.ddaline(d.x2,d.y2,d.x1,d.y2);
    d.ddaline(d.x1,d.y2,d.x1,d.y1);
    d.xmid=abs((d.x1+d.x2))/2;
    d.ymid=abs((d.y1+d.y2))/2;
    d.ddaline(d.xmid,d.y1,d.x2,d.ymid);// (x1,y1) and (x2,y1)
    d.ddaline(d.x2,d.ymid,d.xmid,d.y2);
    d.ddaline(d.xmid,d.y2,d.x1,d.ymid);
    d.ddaline(d.x1,d.ymid,d.xmid,d.y1);
    float rad,cal,sidex,sidey;
    sidex=abs(d.x2-d.x1);
    sidey=abs(d.y2-d.y1);
    cal=pow(sidex,2)+pow(sidey,2);
    cal=2*sqrt(cal);
    rad=(sidex*sidey)/cal;
    cout<<sidex<<" "<<sidey;
```

```
        cout<<" "<<rad;

        d.bdrawcircle(d.xmid,d.ymid,rad);

        getch();

        closegraph();

        // getch();

        return 0;

}
```



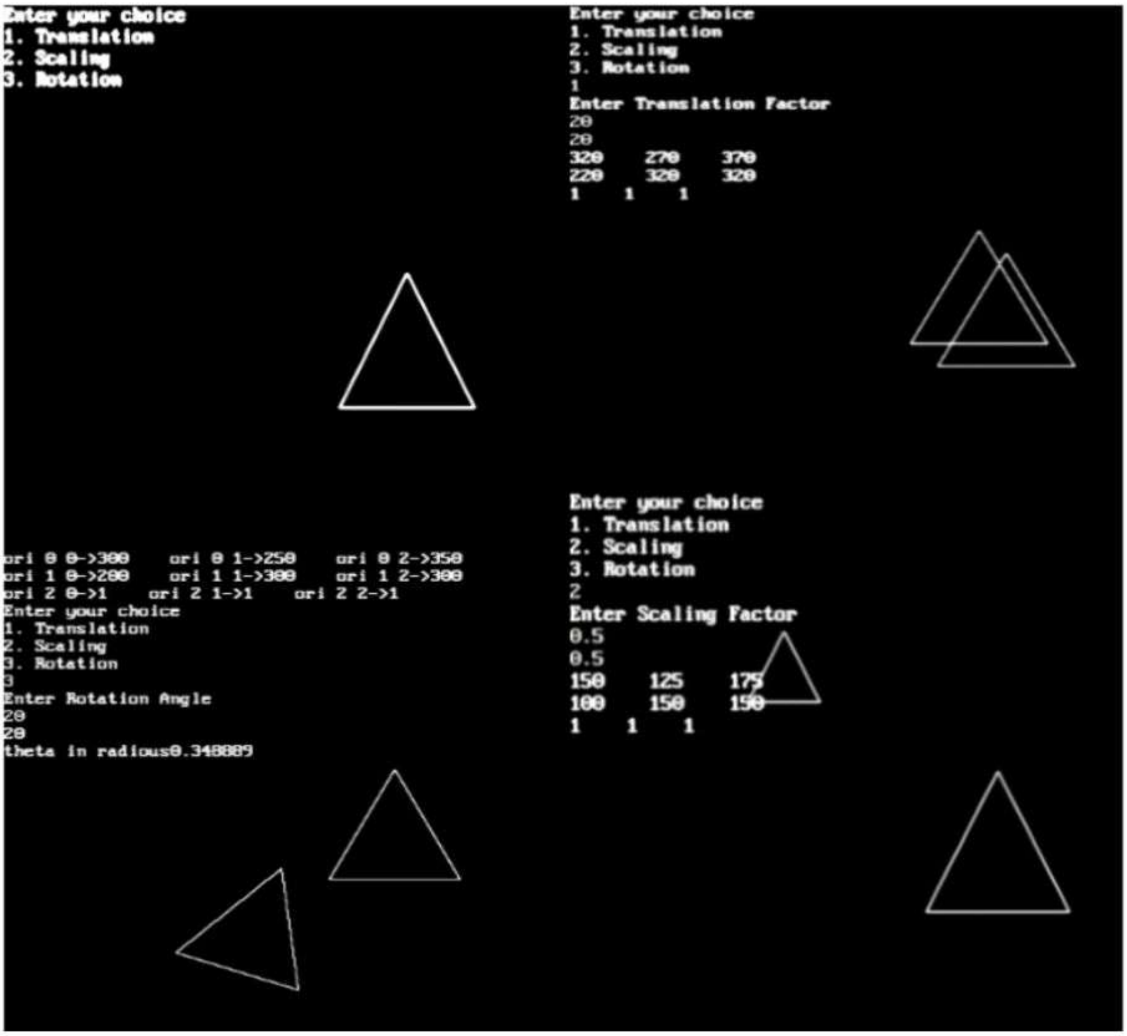
```
Enter x1100
Enter y1100
Enter x2200
Enter y2200
100 100 35.355339
```

```
#include<iostream>
#include<conio.h>
#include<graphics.h>
#include<stdlib.h>
#include<stdio.h>
#include<math.h>
using namespace std;
class trans
{
	public:
	float transco[3][3];
	// float orico[3][3];
	float scalco[3][3];
	float rotco[3][3];
	void drawtri(float [3][3]);
	void translation(int,int,float [3][3]);
	void scaling(float,float,float [3][3]);
	void rotation(float,float [3][3]);
};
void trans::drawtri(float co[3][3])
{
//clrscr();
	line(co[0][0],co[1][0],co[0][1],co[1][1]);
	line(co[0][1],co[1][1],co[0][2],co[1][2]);
	line(co[0][2],co[1][2],co[0][0],co[1][0]);
}
void trans::translation(int tx,int ty,float orico[3][3])
{
	cout<<"Enter Translation Factor"<<endl;
```

```
    cin>>tx>>ty;
    int i,j;
    for(i=0;i<3;i++)
    {
        transco[0][i]=orico[0][i]+tx;
        transco[1][i]=orico[1][i]+ty;
        transco[2][i]=1;
    }
    for(i=0;i<3;i++)
    {
        for(j=0;j<3;j++)
        {
            cout<<transco[i][j]<<" ";
        }
    cout<<endl;
    }
}
void trans::scaling(float sx,float sy,float orico[3][3])
{
    cout<<"Enter Scaling Factor"<<endl;
    cin>>sx>>sy;
    int i,j;
    for(i=0;i<3;i++)
    {
        scalco[0][i]=orico[0][i]*sx;
        scalco[1][i]=orico[1][i]*sy;
        scalco[2][i]=1;
    }
    for(i=0;i<3;i++)
```

```
        {
                for(j=0;j<3;j++)
                {
                        cout<<scalco[i][j]<<" ";
                }
        cout<<endl;
        }
}
void trans::rotation(float theta,float orico[3][3])
{
        cout<<"Enter Rotation Angle"<<endl;
        cin>>theta;
        cout<<theta<<endl;
        theta= theta*(3.14/180);
        cout<<"theta in radious"<<theta<<endl;
        int i,j,refx,refy;
        for(i=0;i<3;i++)
        {
                for(j=0;j<3;j++)
                {
                        rotco[i][j]=0;
                }
        }
        for(i=0;i<3;i++)
        {
                rotco[0][i]=orico[0][i]*cos(theta)-orico[1][i]*sin(theta);
                rotco[1][i]=orico[0][i]*sin(theta)+orico[1][i]*cos(theta);
        }
}
```

```
int main()
{
//clrscr();
    initwindow(800,800);
    int c;
    //int gd= DETECT, gm;
    //initgraph(&gd,&gm,"C:\\TurboC3\\BGI");
    trans t;
    int tx,ty;
    float sx,sy;
    float theta;
    float orico[3][3] ={{300,250,350},{200,300,300},{1,1,1}};
    for(int i=0;i<3;i++)
    {
        for(int j=0;j<3;j++)
        {
            cout<<"ori"<<" "<<i<<" "<<j<<"->"<<orico[i][j]<<" ";
        }
        cout<<endl;
    }
    t.drawtri(orico);
    cout<<"Enter your choice"<<endl;
    cout<<"1. Translation"<<endl;
    cout<<"2. Scaling"<<endl;
    cout<<"3. Rotation"<<endl;
    cin>>c;
    switch(c)
    {
        case 1:
                t.translation(tx,ty,orico);

                t.drawtri(t.transco);

                break;

                case 2:

                t.scaling(sx,sy,orico);

                t.drawtri(t.scalco);

                break;

                case 3:

                t.rotation(theta,orico);

                t.drawtri(t.rotco);

                break;

                default:

                cout<<("You have written wrong Choice");

        }

        getch();

        return 0;

}
```



```
Enter your choice
1. Translation
2. Scaling
3. Rotation
```

```
Enter your choice
1. Translation
2. Scaling
3. Rotation
1
Enter Translation Factor
20
20
320     270     370
220     320     320
1    1    1
```

```
ori 0 0->300    ori 0 1->250    ori 0 2->350
ori 1 0->200    ori 1 1->300    ori 1 2->300
ori 2 0->1    ori 2 1->1    ori 2 2->1
Enter your choice
1. Translation
2. Scaling
3. Rotation
3
Enter Rotation Angle
20
20
theta in radious0.348889
```

```
Enter your choice
1. Translation
2. Scaling
3. Rotation
2
Enter Scaling Factor
0.5
0.5
150    125    175
100    150    150
1    1    1
```