Name: Shraddha Rajkumar Kotwar

Std: SE(COMP)

Div: B

Roll no: 19

Subject: OOPCGL

Assignment No : 1

Problem Statement:

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

Program:

```
#include<iostream.h>

#include<graphics.h>

#include<stdlib.h>

#include<conio.h>

#include<dos.h>


struct edge


{

int x1,y1,x2,y2,flag;

};


int main()
```

```
{
int n,i,j,k,gd,gm,x[10],y[10],ymax=0,ymin=480,yy,temp;
struct edge ed[10],temped; //ed[3].x1,ed[3].y1
float dx,dy,m[10],x_int[10],inter_x[10];
initgraph(&gd,&gm,"c://Turboc3//BGI");

cout<<"\n Enter the number of vertices of the graph: "; cin>>n;
cout<<"\n Enter the vertices: \n";
for(i=0;i<n;i++)
{
cout<<"x"<<i<<":"; cin>>x[i];
cout<<"y"<<i<<":"; cin>>y[i];
if(y[i]>ymax)
ymax=y[i];
if(y[i]<ymin)
ymin=y[i];
ed[i].x1=x[i]; //ed[0].x1=x[0] ed[0].y1=y[0];
ed[i].y1=y[i];

}

for(i=0;i<n-1;i++) //store the edge information
{
ed[i].x2=ed[i+1].x1; //ed[0].x2=ed[1].x1;
```

```
ed[i].y2=ed[i+1].y1;
ed[i].flag=0;
}
ed[i].x2=ed[0].x1; //i=n-1
ed[i].y2=ed[0].y1;
ed[i].flag=0;

for(i=0;i<n-1;i++) //check for y1>y2 if not interchange it
{
if(ed[i].y1<ed[i].y2)
{
temp=ed[i].x1;
ed[i].x1=ed[i].x2;
ed[i].x2=temp;
temp=ed[i].y1;
ed[i].y1=ed[i].y2;
ed[i].y2=temp;
}
}
/*for(i=0;i<n;i++) //draw polygon
{
line(ed[i].x1,ed[i].y1,ed[i].x2,ed[i].y2);
} */

for(i=0;i<n-1;i++) //storing the edges as y1,y2,x1
```

```
{
for(j=0;j<n-1;j++)
{
if(ed[j].y1<ed[j+1].y1)
{
temped=ed[j];
ed[j]=ed[j+1];

ed[j+1]=temped;
}
if(ed[j].y1==ed[j+1].y1)
{
if(ed[j].y2<ed[j+1].y2)
{
temped=ed[j];
ed[j]=ed[j+1];
ed[j+1]=temped;
}
if(ed[j].y2==ed[j+1].y2)
{
if(ed[j].x1<ed[j+1].x1)
{
temped=ed[j];
ed[j]=ed[j+1];
ed[j+1]=temped;
```

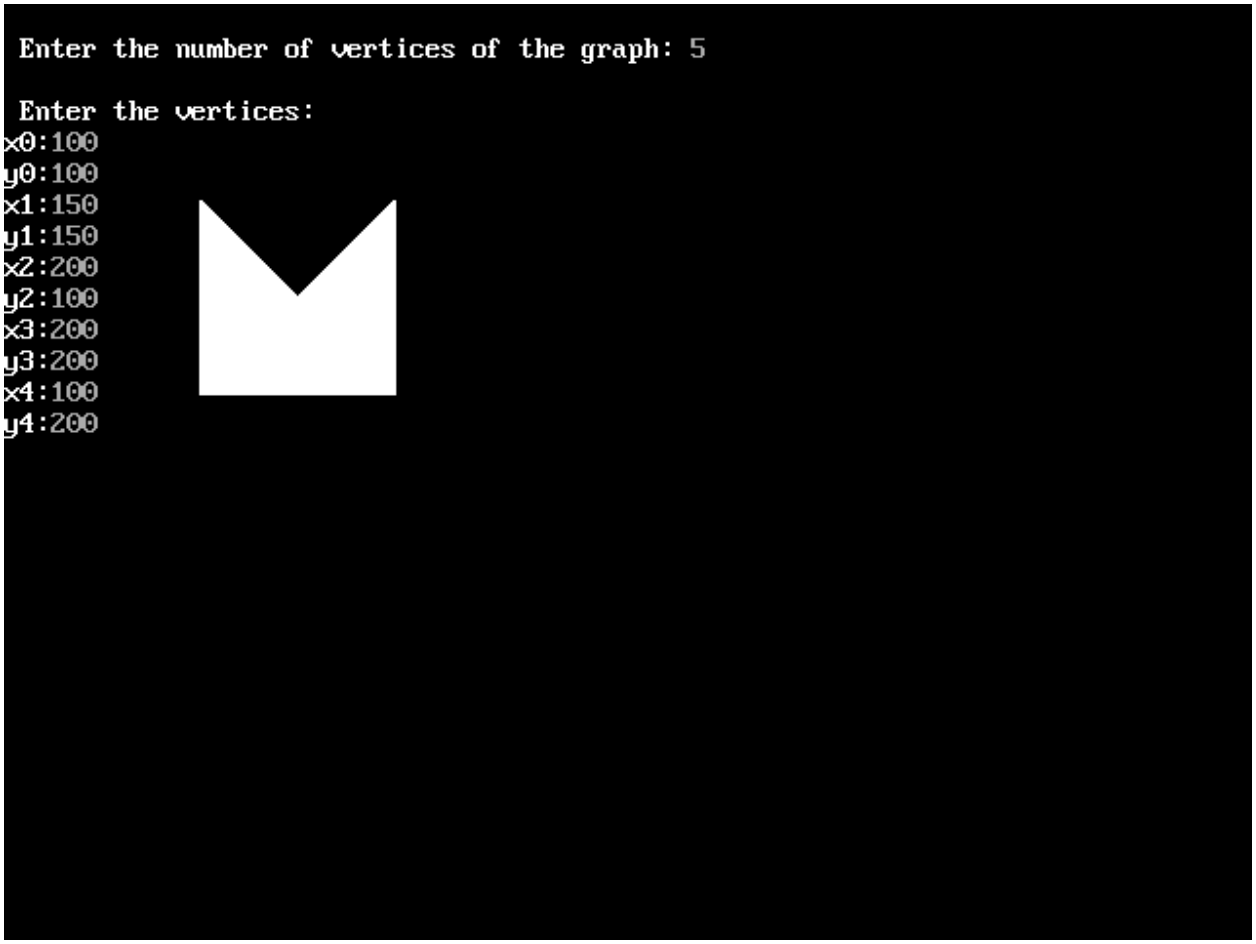
```
}
}
}
}
}

for(i=0;i<n;i++) //calculate 1/slope
{
dx=ed[i].x2-ed[i].x1;
dy=ed[i].y2-ed[i].y1;
if(dy==0)
m[i]=0;
else
m[i]=dx/dy;
inter_x[i]=ed[i].x1;

}
yy=ymax;

while(yy>ymin) //Mark active edges
{
for(i=0;i<n;i++)
{
if(yy>ed[i].y2 && yy<=ed[i].y1 && ed[i].y1!=ed[i].y2)
ed[i].flag=1;
```

```
else
ed[i].flag=0;
}

j=0;
for(i=0;i<n;i++) //Finding x intersections
{
if(ed[i].flag==1)
{
if(yy==ed[i].y1)
{
x_int[j]=ed[i].x1;
j++;
/*if(ed[i-1].y1==yy&&ed[i-1].y1<yy)
{
x_int[j]=ed[i].x1;
j++;
}
if(ed[i+1].y1==yy&&ed[i+1].y1<yy)
{
x_int[j]=ed[i].x1;
j++;
} */

}
```

```
else
{

x_int[j]=inter_x[i]+(-m[i]);
inter_x[i]=x_int[j];
j++;
}
}
}

for(i=0;i<j;i++) //sorting the x intersections
{
for(k=0;k<j-1;k++)
{
if(x_int[k]>x_int[k+1])
{
temp=x_int[k];
x_int[k]=x_int[k+1];
x_int[k+1]=temp;
}
}
}

for(i=0;i<j;i+=2) //Extracting x values to draw a line
```

```
{
line(x_int[i],yy,x_int[i+1],yy);
delay(100);
}
yy--;
} //end of while loop
delay(3000);
getch();
closegraph();
return 0;

}
```

Output:



Assignment No : 2

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

Program:

```
#include<iostream.h>

#include<dos.h>

#include<stdlib.h>

#include<math.h>

#include<graphics.h>
```

```
/* Defining structure for end point of line */

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()

{

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)
```

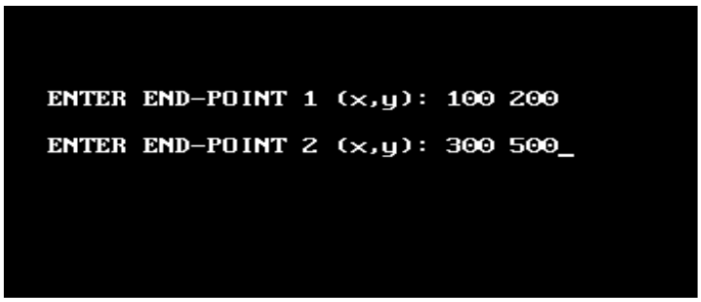
```
ptemp.code[0]='1'; /* TOP */

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++)
```

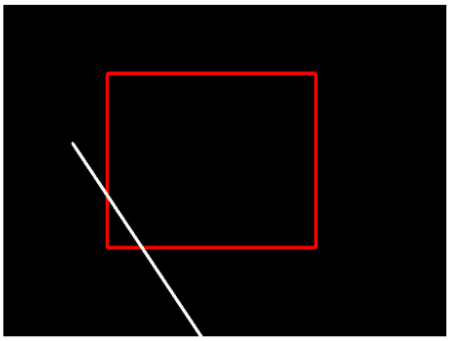
```
{
if((p1.code[i]!='0')||(p2.code[i]!='0'))

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;
```

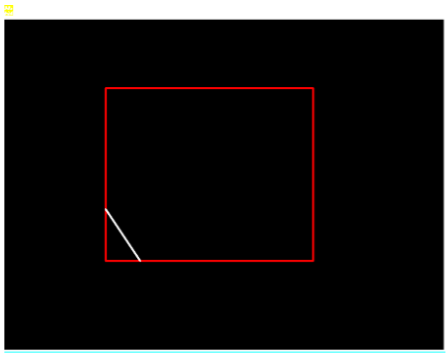
```
float m,k;
if( p1.code[3]=='1') /* Cutting LEFT Edge */
x=150;
if(p1.code[2]=='1') /* Cutting RIGHT Edge */
x=450;
if((p1.code[3]=='1')||(p1.code[2]=='1'))
{
m=(float) (p2.y-p1.y)/(p2.x-p1.x);
k=(p1.y+(m*(x-p1.x)));
temp.y=k;
temp.x=x;
if(temp.y<=350&&temp.y>=100)
return(temp);

}
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);

}
```

Output:







Assignment No: 3

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

Program:

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

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();
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;

}
```

Output:

Assignment No: 4

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

Program:

```
#include<iostream.h>

#include<conio.h>

#include<graphics.h>

#include<stdlib.h>
```

```
#include<math.h>
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])
```

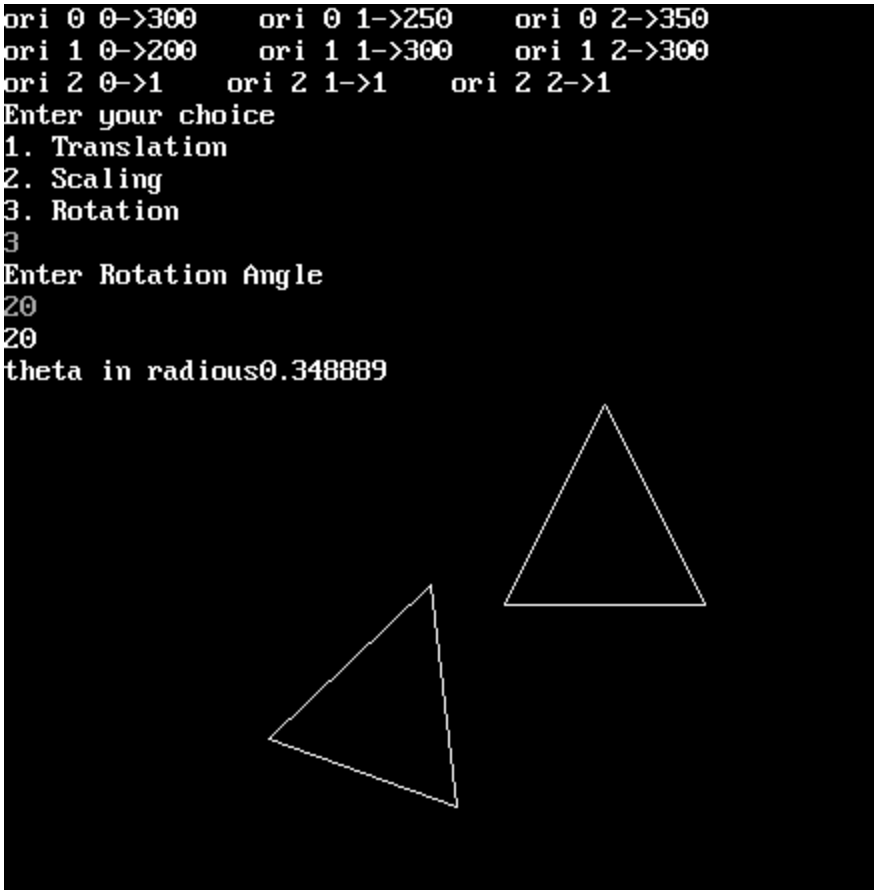
```
{

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);
}

}
void main()
{
clrscr();
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();


}
```



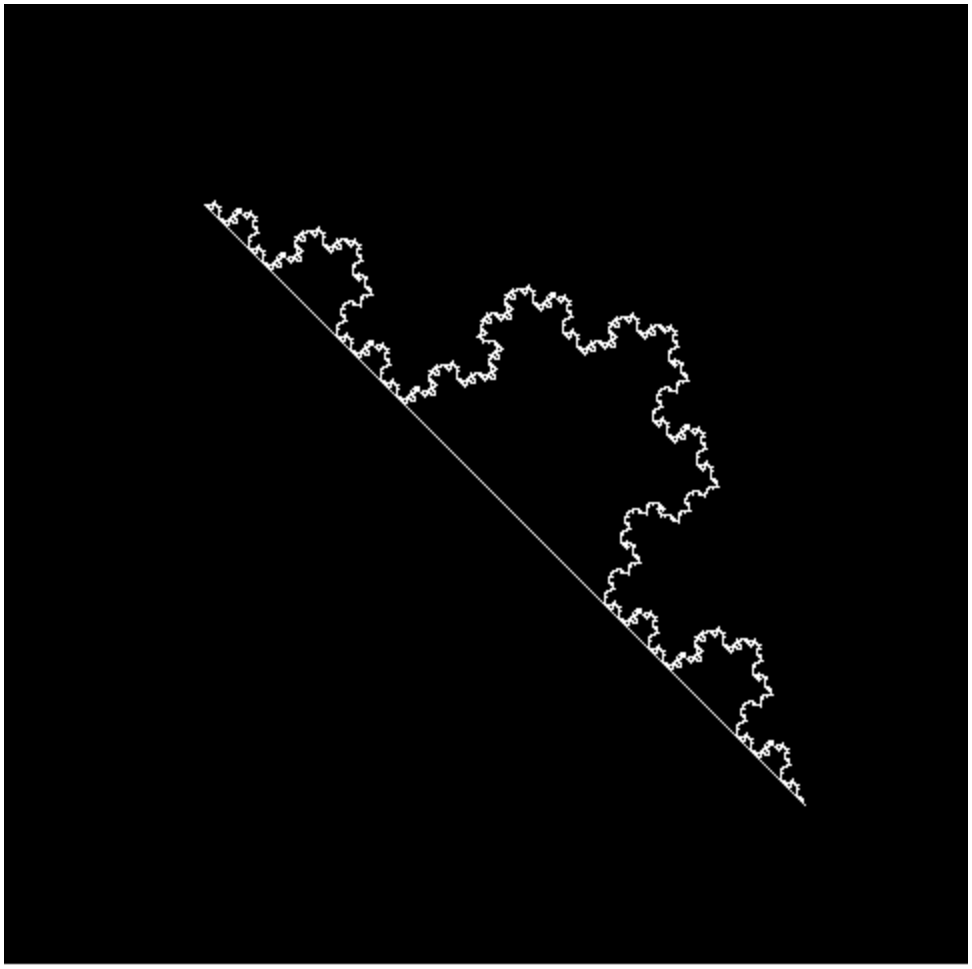
Assignment No: 5

Problem Statement: Write C++ program to generate fractal patterns by using Koch curves.

```
#include<stdio.h>

#include<conio.h>

#include<math.h>

#include<graphics.h>

#include<dos.h>
```

```
void koch(int x1,int y1,int x2,int y2,int it){
float ang=60*M_PI/180;
int x3=(2*x1+x2)/3;
int y3=(2*y1+y2)/3;

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

int x= x3+(x4-x3)*cos(ang)+(y4-y3)*sin(ang);
int y= y3-(x4-x3)*sin(ang)+(y4-y3)*cos(ang);

if(it>0)
{
koch(x1,y1,x3,y3,it-1);
koch(x3,y3,x,y,it-1);
koch(x,y,x4,y4,it-1);
koch(x4,y4,x2,y2,it-1);
}
else{
//delay(100);
line(x1,y1,x3,y3);
//delay(100);
line(x3,y3,x,y);
//delay(100);
```

```
line(x,y,x4,y4);
//delay(100);
line(x4,y4,x2,y2);
//delay(100);
}

}
int main()
{
int gd = DETECT,gm;
initgraph(&gd,&gm,"c:\\TURBOC3\\BGI");
int x1=100,y1=100,x2=400,y2=400;

line(100,100,400,400);
//delay(50);
koch(x1,y1,x2,y2,5);
getch();
return 0;
}
```



Assignment No : 6

Problem Statement: Write OpenGL program to draw Sun Rise and Sunset.

Assignment No: 7

Problem Statement: Write C++ program to draw man walking in the rain with an umbrella. Apply the concept of polymorphism.

```
#include<conio.h>
#include<iostream.h>
#include<graphics.h>
#include<stdlib.h>
#include<dos.h>
void main(){
int gd=DETECT,gm;
initgraph(&gd,&gm,"C:\\TurboC3\\BGI");
int xmov,x,y;

//xmov=10;
for(xmov=1;xmov<200;xmov=xmov+5)
{
line(0,400,639,400);
circle(30+xmov,280,20); //head
line(30+xmov,300,30+xmov,350);//body
line(30+xmov,330,70+xmov,330); //hand
if(xmov%2==0)
{

line(30+xmov,350,25+xmov,400); //left leg
line(30+xmov,350,10+xmov,400); //right leg
}
```

```
else
{
line(30+xmov,350,25+xmov,400);
delay(25);
}
line(70+xmov,250,70+xmov,330); //umbrella
pieslice(80+xmov,250,180,0,80);
for(int i=0;i<=300;i++)
{
x=random(800);
y=random(800);
outtextxy(x,y,"/");

}
delay(600);
cleardevice();
}
getch();
closegraph();

}
```