Name : Dishank Shekokare

Roll no. : 5216

SE I.T

**DDA LINE**

INPUT

#include <stdio.h>

#include <math.h>

#include<windows.h>

#include <GL/glut.h>

double X1, Y1, X2, Y2;

float round\_value(float v)

{

return floor(v + 0.5);

}

void LineDDA(void)

{

double dx=(X2-X1);

double dy=(Y2-Y1);

double steps;

float xInc,yInc,x=X1,y=Y1;

/\* Find out whether to increment x or y \*/

steps=(abs(dx)>abs(dy))?(abs(dx)):(abs(dy));

xInc=dx/(float)steps;

yInc=dy/(float)steps;

/\* Clears buffers to preset values \*/

glClear(GL\_COLOR\_BUFFER\_BIT);

/\* Plot the points \*/

glBegin(GL\_POINTS);

/\* Plot the first point \*/

glVertex2d(x,y);

int k;

/\* For every step, find an intermediate vertex \*/

for(k=0;k<steps;k++)

{

x+=xInc;

y+=yInc;

/\* printf("%0.6lf %0.6lf\n",floor(x), floor(y)); \*/

glVertex2d(round\_value(x), round\_value(y));

}

glEnd();

glFlush();

}

void Init()

{

/\* Set clear color to white \*/

glClearColor(1.0,1.0,1.0,0);

/\* Set fill color to black \*/

glColor3f(0.0,0.0,0.0);

/\* glViewport(0 , 0 , 640 , 480); \*/

/\* glMatrixMode(GL\_PROJECTION); \*/

/\* glLoadIdentity(); \*/

gluOrtho2D(0 , 640 , 0 , 480);

}

int main(int argc, char \*\*argv)

{

printf("Enter two end points of the line to be drawn:\n");

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

printf("\nEnter Point1( X1 , Y1):\n");

scanf("%lf%lf",&X1,&Y1);

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

printf("\nEnter Point1( X2 , Y2):\n");

scanf("%lf%lf",&X2,&Y2);

/\* Initialise GLUT library \*/

glutInit(&argc,argv);

/\* Set the initial display mode \*/

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

/\* Set the initial window position and size \*/

glutInitWindowPosition(0,0);

glutInitWindowSize(640,480);

/\* Create the window with title "DDA\_Line" \*/

glutCreateWindow("DDA\_Line");

/\* Initialize drawing colors \*/

Init();

/\* Call the displaying function \*/

glutDisplayFunc(LineDDA);

/\* Keep displaying untill the program is closed \*/

glutMainLoop();

}

OUTPUT

Enter two end points of the line to be drawn:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Enter Point1( X1 , Y1):

50

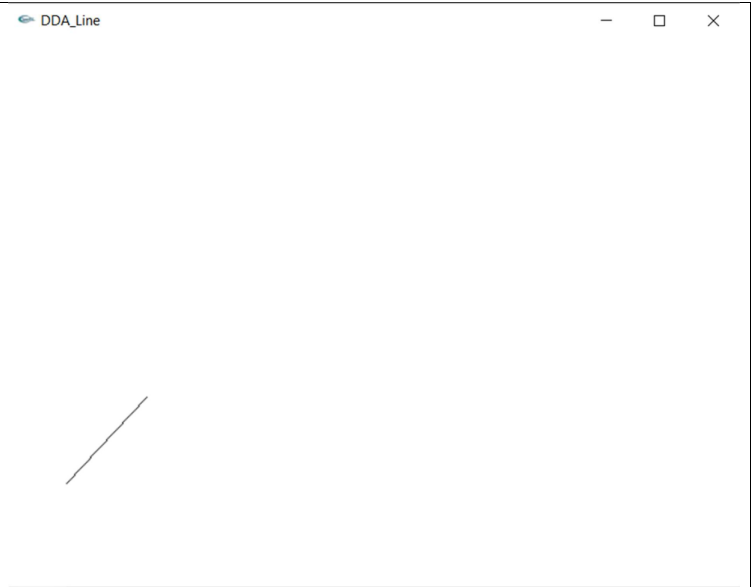
90

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Enter Point1( X2 , Y2):

120

165



Name : Dishank Shekokare

Roll no. : 5216

SE I.T

**Bresenhiem Line Generation**

INPUT

Starting and ending point of line

#include<windows.h>

#include <GL/glut.h>

#include<math.h>

#include<stdlib.h>

#include<iostream>

#include<stdio.h>

using namespace std;

int sign(float arg)

{

if(arg<0)

return -1;

else if(arg==0)

return 0;

else

return 1;

}

void Bre(int X1,int Y1,int X2,int Y2)

{

float dx=abs(X2-X1);

float dy=abs(Y2-Y1);

int s1,s2,exc,y,x,i;

float g,temp;

x=X1;

y=Y1;

s1=sign(X2-X1);

s2=sign(Y2-Y1);

glBegin(GL\_POINTS);

if(dy>dx)

{

temp=dx;

dx=dy;

dy=temp;

exc=1;

}

else

{

exc=0;

}

g=2\*dy-dx;

i=1;

while(i<=dx)

{

glVertex2d(x,y);

while(g>=0)

{

if(exc==1)

x=x+s1;

else

y=y+s2;

g=g-2\*dx;

}

if(exc==1)

y=y+s2;

else

x=x+s1;

g=g+2\*dy;

i++;

}

glEnd();

glFlush();

}

void display()

{

Bre(40,40,200,200);

Bre(250,4000,100,100);

}

void myInit(void)

{ glClearColor(1.0,1.0,1.0,0.0);

glColor3f(1,1,1);

glPointSize(1.0);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0.0,640.0,0.0,480.0);

}

int main(int argc,char \*\*argv)

{

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB);

glutInitWindowSize(640,480);

glutInitWindowPosition(100,150);

glutCreateWindow("Bresenham Line ");

glutDisplayFunc(display);

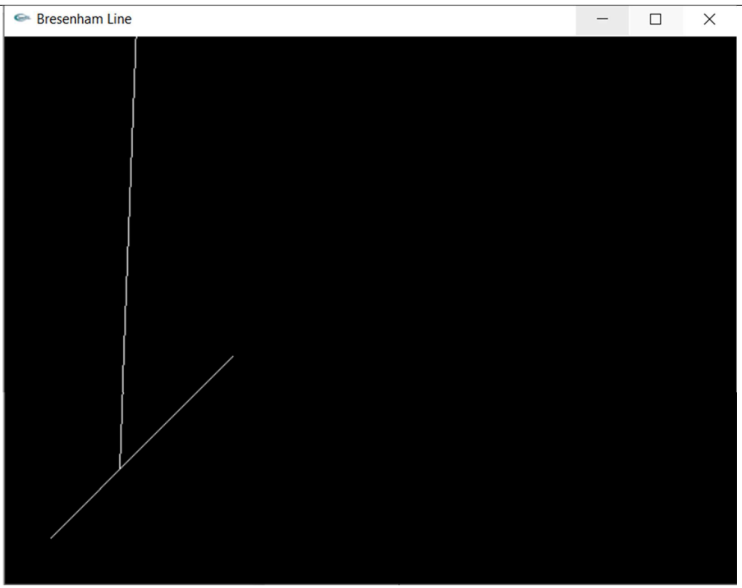
myInit();

glutMainLoop();

return 0;

}

OUTPUT



Name : Dishank Shekokare

Roll no. : 5216

SE I.T

**Bresenham’s Circle Drawing:**

INPUT

#include<windows.h>

#include <GL/glut.h>

#include<math.h>

#include<stdlib.h>

#include<iostream>

#include<stdio.h>

using namespace std;

void symmetry(double xc,double yc,double x,double y)

{

glBegin(GL\_POINTS);

glVertex2i(xc+x, yc+y);

glVertex2i(xc+x, yc-y);

glVertex2i(xc+y, yc+x);

glVertex2i(xc+y, yc-x);

glVertex2i(xc-x, yc-y);

glVertex2i(xc-y, yc-x);

glVertex2i(xc-x, yc+y);

glVertex2i(xc-y, yc+x);

glEnd();

}

void circle(double x1,double y1,double r)

{

int x=0,y=r;

float pk=(5.0/4.0)-r;

/\* Plot the points \*/

/\* Plot the first point \*/

symmetry(x1,y1,x,y);

int k;

/\* Find all vertices till x=y \*/

while(x < y)

{

x = x + 1;

if(pk < 0)

pk = pk + 2\*x+1;

else

{

y = y - 1;

pk = pk + 2\*(x - y) + 1;

}

symmetry(x1,y1,x,y);

}

glFlush();

}

void display()

{

circle(200,200,120);

}

void myInit(void)

{ glClearColor(1.0,1.0,1.0,0.0);

glColor3f(1,1,1);

glPointSize(2.0);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0.0,640.0,0.0,480.0);

}

int main(int argc,char \*\*argv)

{

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB);

glutInitWindowSize(640,480);

glutInitWindowPosition(100,150);

glutCreateWindow("Circle");

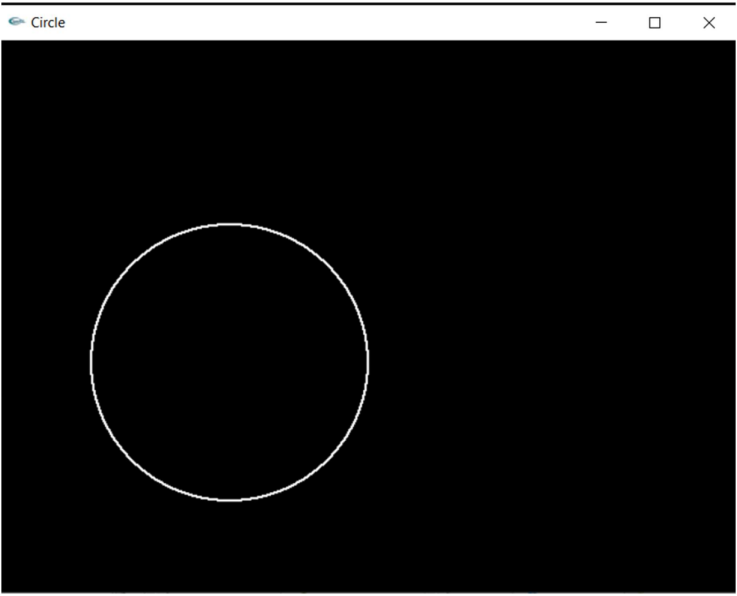
glutDisplayFunc(display);

myInit();

glutMainLoop();

return 0;

}

OUTPUT   


Name : Dishank Shekokare

Roll no. : 5216

SE I.T

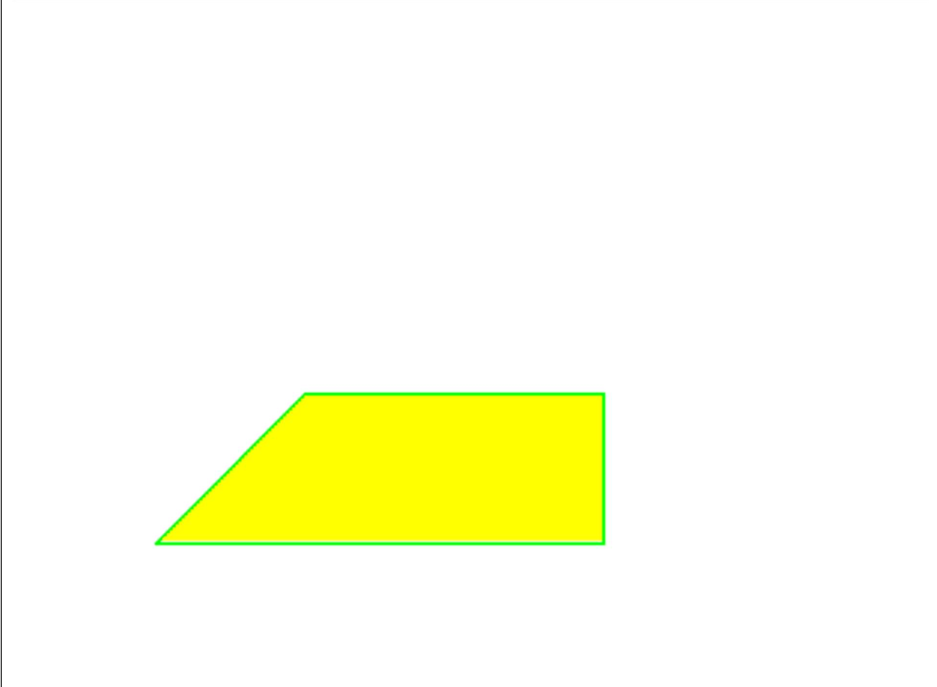
**Seed Fill And Boundary Fill**

INPUT

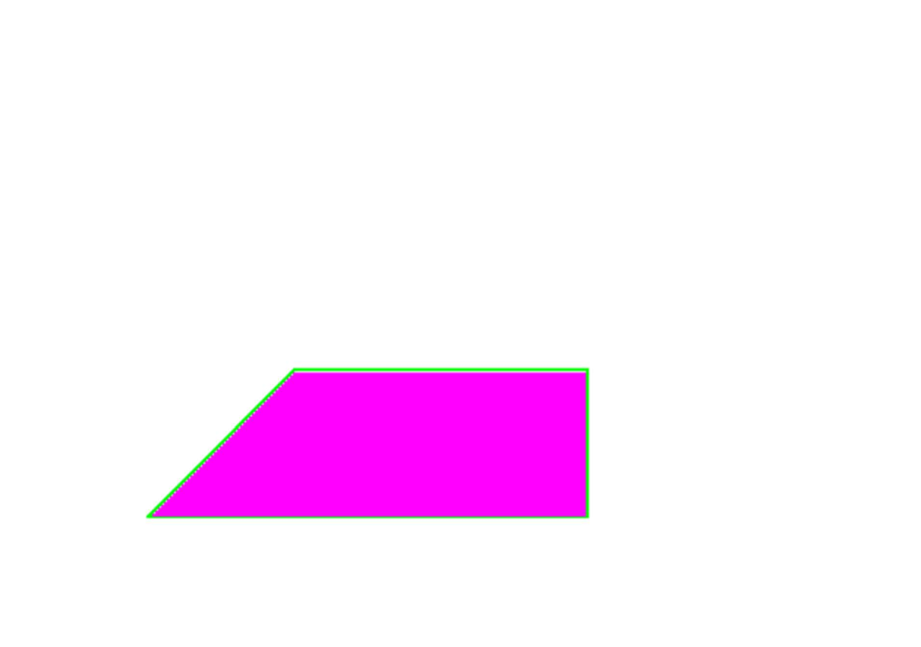
#include<iostream>  
#include<math.h>  
#include<windows.h>  
#include<GL/glut.h>  
using namespace std;  
int a, b, Dx, Dy, temp, interchange, e; //Specifying parameters for algorithms  
int s1, s2;  
void myInit() { //Initialization of viewing parameters  
glClearColor(1.0, 1.0, 1.0, 0.0);  
glMatrixMode(GL\_PROJECTION);  
gluOrtho2D(0.0, 640.0, 0.0, 480.0);  
}  
int sign(int a) { //Function to determine sign of a number  
if (a > 0) {  
return 1;  
}  
else if (a < 0) {  
return -1;  
}  
}  
void plot(int a, int b) { //Function to plot points  
glBegin(GL\_POINTS);  
glVertex2i(a, b);  
glEnd();  
glFlush();  
}  
void DrawPolygon(int x, int y, int m, int n) { //Function to draw polygon using DDA algorithm  
a = x; b = y;  
Dx = abs(m - x);  
Dy = abs(n - y);  
s1 = sign(m - x);  
s2 = sign(n - y);  
if (Dy > Dx) {  
temp = Dx;  
Dx = Dy;  
Dy = temp;  
interchange = 1;  
}  
else {  
}  
interchange = 0;  
e = 2 \* Dy - Dx;  
plot(a, b);  
for (int i = 1; i <= Dx; i++) {  
plot(a, b);  
while (e >= 0) {  
if (interchange == 1)  
a = a + s1;  
else  
b = b + s2;  
e = e - 2 \* Dx;  
}  
if (interchange == 1) {  
b = b + s2;  
}  
else {  
}  
a = a + s1;  
e = e + 2 \* Dy;  
}  
}  
void putpixel(int c, int d, float\* fillColor) { //Function to color polygon  
glColor3f(fillColor[0], fillColor[1], fillColor[2]); //Setting the color buffer  
glBegin(GL\_POINTS);  
glVertex2i(c, d);  
glEnd();  
glFlush();  
}  
void boundaryFill4(int x, int y, float\* fillColor, float\* boundarycolor) { //Boundary fill  
algorithm  
float color[3];  
glReadPixels(x, y, 1.0, 1.0, GL\_RGB, GL\_FLOAT, color);  
if ((color[0] != boundarycolor[0] || color[1] != boundarycolor[1] || color[2] !=  
boundarycolor[2]) && (  
color[0] != fillColor[0] || color[1] != fillColor[1] || color[2] != fillColor[2])) {  
putpixel(x, y, fillColor);  
boundaryFill4(x + 1, y, fillColor, boundarycolor); //Recurssive calls  
boundaryFill4(x - 2, y, fillColor, boundarycolor);  
boundaryFill4(x, y + 2, fillColor, boundarycolor);  
boundaryFill4(x, y - 2, fillColor, boundarycolor);  
}  
}

void Floodfill4(int x, int y, float\* fillColor, float\* interiorColor) { //Floodfill algorithm  
float color[3];  
glReadPixels(x, y, 1.0, 1.0, GL\_RGB, GL\_FLOAT, color);  
if (color[0] == interiorColor[0] && color[1] == interiorColor[1] && color[2] ==  
interiorColor[2]) {  
putpixel(x, y, fillColor);  
}  
else {  
}  
}  
Floodfill4(x + 1, y, fillColor, interiorColor); //Recurssive calls  
Floodfill4(x - 1, y, fillColor, interiorColor);  
Floodfill4(x, y + 1, fillColor, interiorColor);  
Floodfill4(x, y - 1, fillColor, interiorColor);  
return;  
void myMouse(int button, int state, int x, int y) { //Function to select seed point  
y = 480 - y;  
if (button == GLUT\_LEFT\_BUTTON)  
{  
if (state == GLUT\_DOWN)  
{  
float boundaryCol[] = { 0,1,0 };  
float fillcolor[] = { 1,0,1 };  
boundaryFill4(x, y, fillcolor, boundaryCol);  
}  
}  
}  
void myMouse1(int button, int state, int x, int y) {  
y = 480 - y;  
if (button == GLUT\_LEFT\_BUTTON && state == GLUT\_DOWN) {  
float interiorcolor[] = { 1,1,1 };  
float fillcolor[] = { 1,1,0 };  
Floodfill4(x, y, fillcolor, interiorcolor);  
}  
}  
void myDisplay() { //Draws polygon on the screen  
glLineWidth(3.0);  
glPointSize(2.0);  
glClear(GL\_COLOR\_BUFFER\_BIT);  
glColor3f(0, 1, 0);  
DrawPolygon(100, 100, 200, 200);  
DrawPolygon(200, 200, 400, 200);  
DrawPolygon(400, 200, 400, 100);  
DrawPolygon(400, 100, 100, 100);  
glEnd();  
glFlush();  
}  
void myMenu(int item) { //Menu function  
if (item == 1) {  
glutMouseFunc(myMouse);  
}  
else if (item == 2) {  
glutMouseFunc(myMouse1);  
}  
}  
int main(int argc, char\*\* argv) {  
glutInit(&argc, argv);  
glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB); //Initializing display mode for  
window  
glutInitWindowSize(640, 480);  
glutInitWindowPosition(200, 200);  
glutCreateWindow("Polygon Filling");  
glutDisplayFunc(myDisplay);  
glutCreateMenu(myMenu);  
glutAttachMenu(GLUT\_RIGHT\_BUTTON);  
glutAddMenuEntry("Boundary fill",1);  
glutAddMenuEntry("Flood fill",2);  
myInit();  
glutMainLoop();  
return 0;  
}

SEED FILL OUTPUT :



BOUNDARY FILL OUTPUT :



Name : Dishank Shekokare

Roll no. : 5216

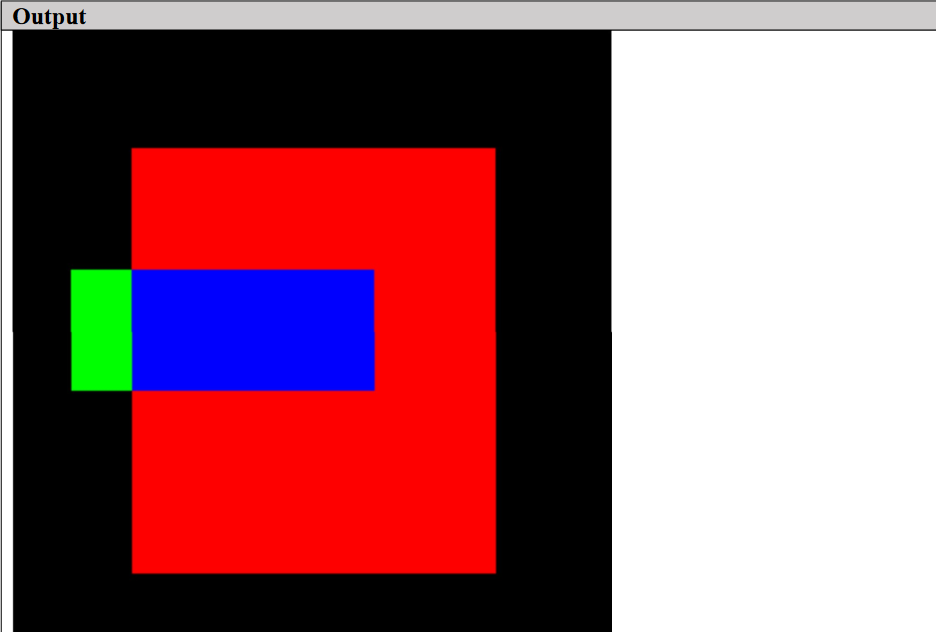
SE I.T

**Line Clipping – Cohen Sutherland**

INPUT

#include <windows.h>  
#include <gl/glut.h>  
struct Point{  
float x,y;  
} w[4],oVer[4];  
int Nout;  
void drawPoly(Point p[],int n){  
glBegin(GL\_POLYGON);  
for(int i=0;i<n;i++)  
glVertex2f(p[i].x,p[i].y);  
glEnd();  
}  
bool insideVer(Point p){  
if((p.x>=w[0].x)&&(p.x<=w[2].x))  
if((p.y>=w[0].y)&&(p.y<=w[2].y))  
return true;  
return false;  
Input  
}  
void addVer(Point p){  
oVer[Nout]=p;  
Nout=Nout+1;  
}  
Point getInterSect(Point s,Point p,int edge){  
Point in;  
float m;  
if(w[edge].x==w[(edge+1)%4].x){ //Vertical Line  
m=(p.y-s.y)/(p.x-s.x);  
in.x=w[edge].x;  
in.y=in.x\*m+s.y;  
}  
else{//Horizontal Line  
m=(p.y-s.y)/(p.x-s.x);  
in.y=w[edge].y;  
in.x=(in.y-s.y)/m;  
}  
return in;  
}  
void clipAndDraw(Point inVer[],int Nin){  
Point s,p,interSec;  
for(int i=0;i<4;i++)  
{  
Nout=0;  
s=inVer[Nin-1];  
for(int j=0;j<Nin;j++)  
{  
p=inVer[j];  
if(insideVer(p)==true){  
if(insideVer(s)==true){  
addVer(p);  
}  
else{  
interSec=getInterSect(s,p,i);  
addVer(interSec);  
addVer(p);  
}  
}  
else{  
if(insideVer(s)==true){  
interSec=getInterSect(s,p,i);  
addVer(interSec);  
}  
}  
s=p;  
}  
inVer=oVer;  
Nin=Nout;  
}  
drawPoly(oVer,4);  
}  
void init(){  
glClearColor(0.0f,0.0f,0.0f,0.0f);  
glMatrixMode(GL\_PROJECTION);  
glLoadIdentity();  
glOrtho(0.0,100.0,0.0,100.0,0.0,100.0);  
glClear(GL\_COLOR\_BUFFER\_BIT);  
w[0].x =20,w[0].y=10;  
w[1].x =20,w[1].y=80;  
w[2].x =80,w[2].y=80;  
w[3].x =80,w[3].y=10;  
}  
void display(void){  
Point inVer[4];  
init();  
// As Window for Clipping  
glColor3f(1.0f,0.0f,0.0f);  
drawPoly(w,4);  
// As Rect  
glColor3f(0.0f,1.0f,0.0f);  
inVer[0].x =10,inVer[0].y=40;  
inVer[1].x =10,inVer[1].y=60;  
inVer[2].x =60,inVer[2].y=60;  
inVer[3].x =60,inVer[3].y=40;  
drawPoly(inVer,4);  
// As Rect  
glColor3f(0.0f,0.0f,1.0f);  
clipAndDraw(inVer,4);  
// Print  
glFlush();  
}  
int main(int argc,char \*argv[]){  
glutInit(&argc,argv);  
glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB);  
glutInitWindowSize(400,400);  
glutInitWindowPosition(100,100);  
glutCreateWindow("Polygon Clipping!");  
glutDisplayFunc(display);  
glutMainLoop();  
return 0;  
}

OUTPUT



Name : Dishank Shekokare

Roll no. : 5216

SE I.T

**2D Transformations**

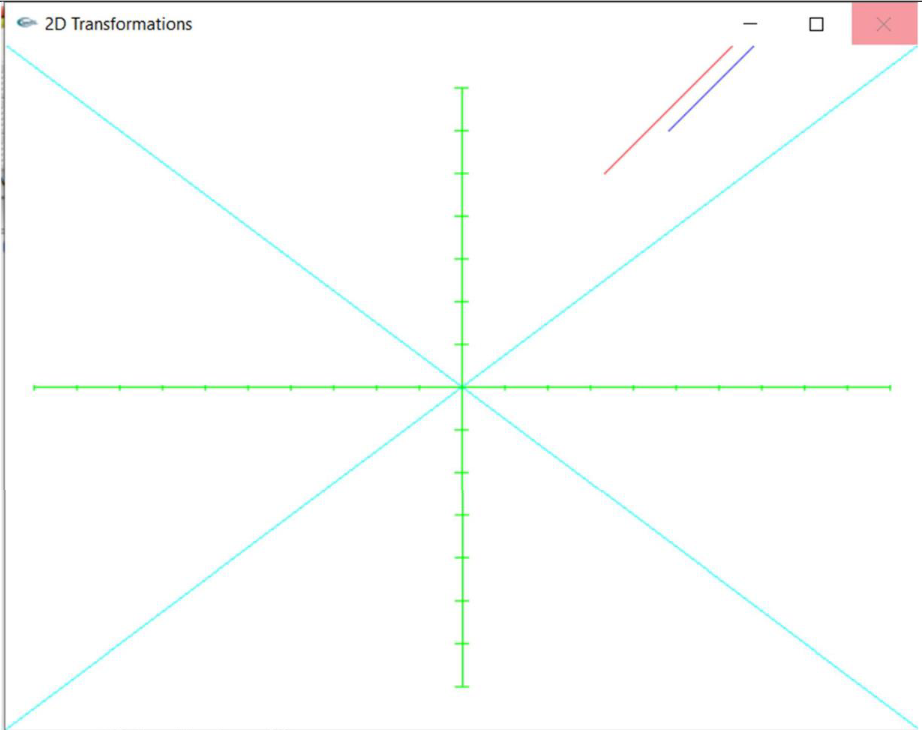
INPUT

Enter an object of any shape for transformation

OUTPUT

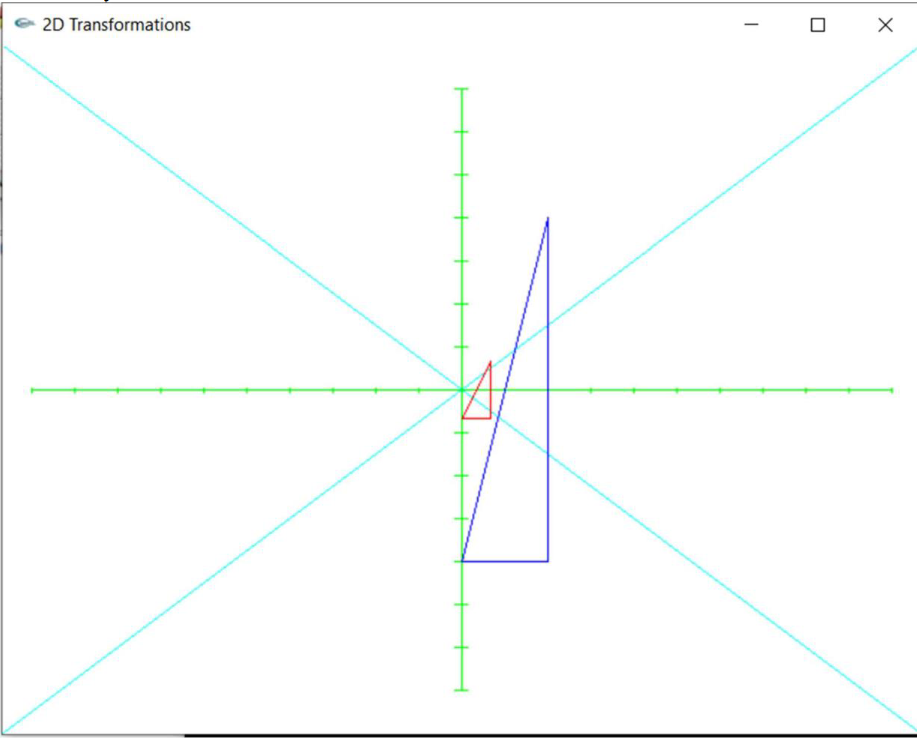
**Translation**

\*\*\*\*\*\*\*\*\*\*\* 2D Transformations \*\*\*\*\*\*\*\*\*  
Enter the no of vertices of polygon : 2  
Enter the coordinate :  
x1,y1 : 100  
150  
x2,y2 : 200  
250  
\*\*\*\*\*\*\*\*\*\*\*MENU\*\*\*\*\*\*\*\*\*\*  
1) Translation  
2) Scaling  
3) Rotation  
4) Rotation of arbitary  
5) Shearing  
6) Reflection  
Enter your choice : 1  
The polygon before translation  
Enter the tx : 45  
Enter the ty : 30



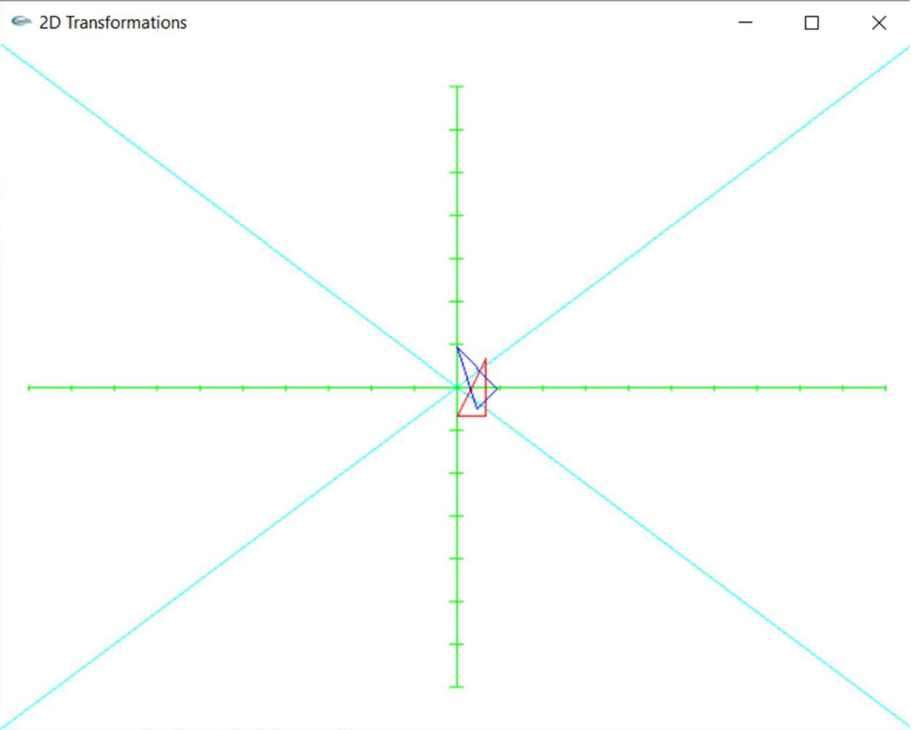
**Scaling**

\*\*\*\*\*\*\*\*\*\*\* 2D Transformations \*\*\*\*\*\*\*\*\*  
Enter the no of vertices of polygon : 3  
Enter the coordinate :  
x1,y1 : 0  
-20  
x2,y2 : 20  
-20  
x3,y3 : 20  
20  
\*\*\*\*\*\*\*\*\*\*\*MENU\*\*\*\*\*\*\*\*\*\*  
1) Translation  
2) Scaling  
3) Rotation  
4) Rotation of arbitary  
5) Shearing  
6) Reflection  
Enter your choice : 2  
The polygon before scaling  
Enter the sx3  
Enter the sy6



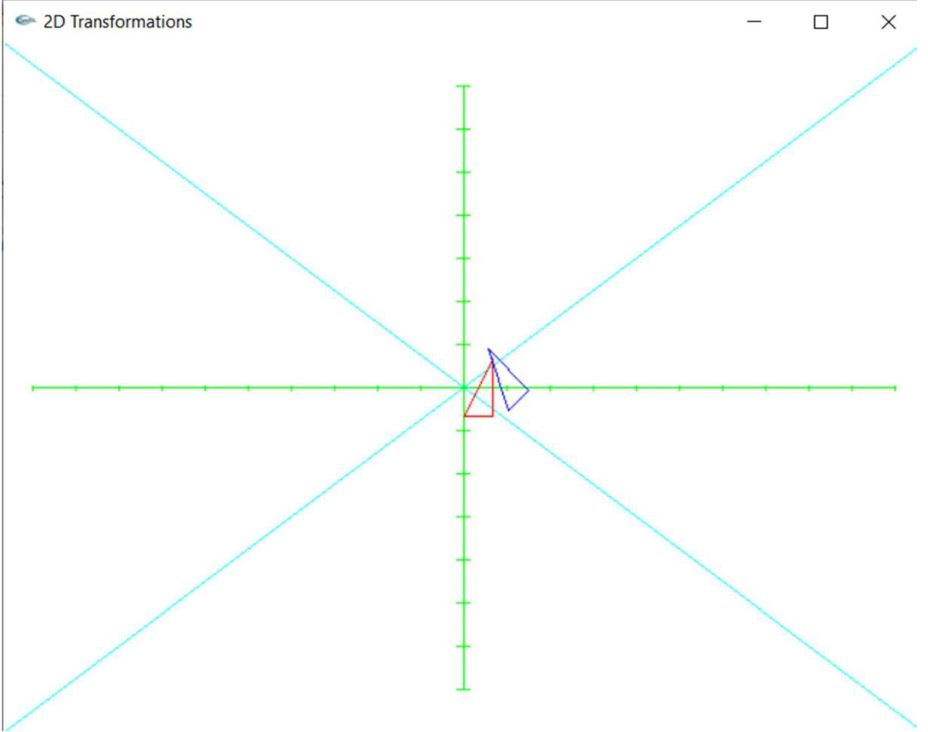
**Rotation**

\*\*\*\*\*\*\*\*\*\*\* 2D Transformations \*\*\*\*\*\*\*\*\*  
Enter the no of vertices of polygon : 3  
Enter the coordinate :  
x1,y1 : 0  
-20  
x2,y2 : 20  
-20  
x3,y3 : 20  
20  
\*\*\*\*\*\*\*\*\*\*\*MENU\*\*\*\*\*\*\*\*\*\*  
1) Translation  
2) Scaling  
3) Rotation  
4) Rotation of arbitary  
5) Shearing  
6) Reflection  
Enter your choice : 3  
The polygon befor rotation  
Enter the angle : 45  
Press 1 for anticlockwise and 2 for clockwise : 1



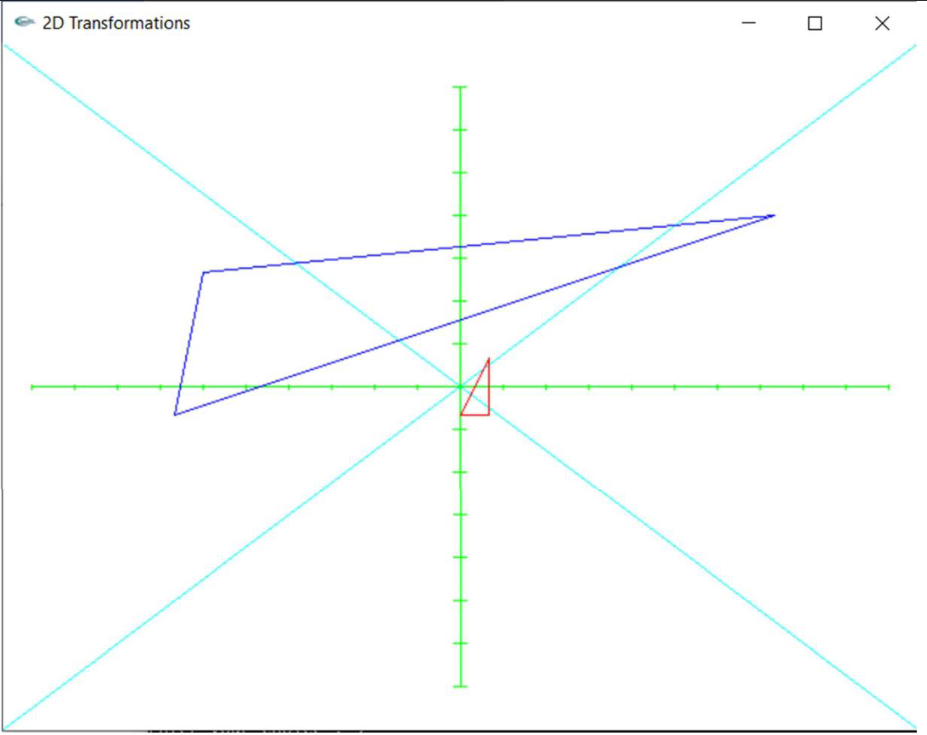
**Rotation of arbitrary**

\*\*\*\*\*\*\*\*\*\*\* 2D Transformations \*\*\*\*\*\*\*\*\*  
Enter the no of vertices of polygon : 3  
Enter the coordinate :  
x1,y1 : 0  
-20  
x2,y2 : 20  
-20  
x3,y3 : 20  
20  
\*\*\*\*\*\*\*\*\*\*\*MENU\*\*\*\*\*\*\*\*\*  
1) Translation  
2) Scaling  
3) Rotation  
4) Rotation of arbitary  
5) Shearing  
6) Reflection  
Enter your choice : 4  
The polygon befor rotation  
Enter the angle : 45  
Press 1 for anticlockwise and 2 for clockwise : 1  
Enter the x and y coordinate : 10



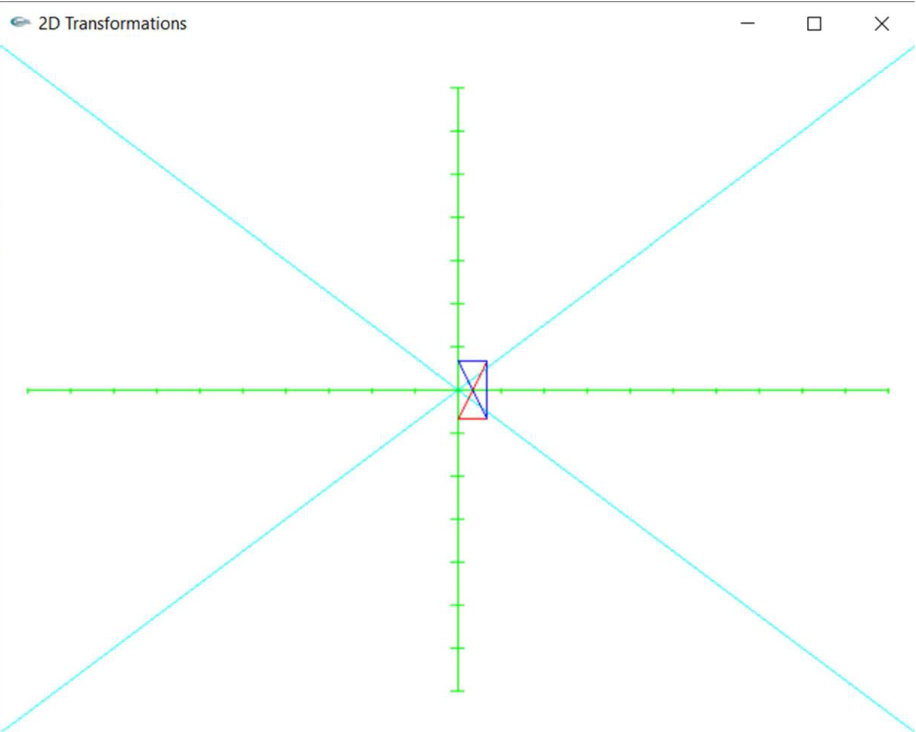
**Shearing**

\*\*\*\*\*\*\*\*\*\*\* 2D Transformations \*\*\*\*\*\*\*\*\*  
Enter the no of vertices of polygon : 3  
Enter the coordinate :  
x1,y1 : 0  
-20  
x2,y2 : 20  
-20  
x3,y3 : 20  
20  
\*\*\*\*\*\*\*\*\*\*\*MENU\*\*\*\*\*\*\*\*\*\*  
1) Translation  
2) Scaling  
3) Rotation  
4) Rotation of arbitary  
5) Shearing  
6) Reflection  
Enter your choice : 5  
The polygon before Shearing  
Enter the Shx : 5  
Enter the Shy : 10



**Reflection**

\*\*\*\*\*\*\*\*\*\*\* 2D Transformations \*\*\*\*\*\*\*\*\*  
Enter the no of vertices of polygon : 3  
Enter the coordinate :  
x1,y1 : 0  
-20  
x2,y2 : 20  
-20  
x3,y3 : 20  
20  
\*\*\*\*\*\*\*\*\*\*\*MENU\*\*\*\*\*\*\*\*\*\*  
1) Translation  
2) Scaling  
3) Rotation  
4) Rotation of arbitary  
5) Shearing  
6) Reflection  
Enter your choice : 6  
The polygon before Reflection  
1. Against X-axis  
2. Against Y-axis  
3. Against Origin  
4. X = Y  
5. X = -Y  
Enter you Choice : 1



Name : Dishank Shekokare

Roll no. : 5216

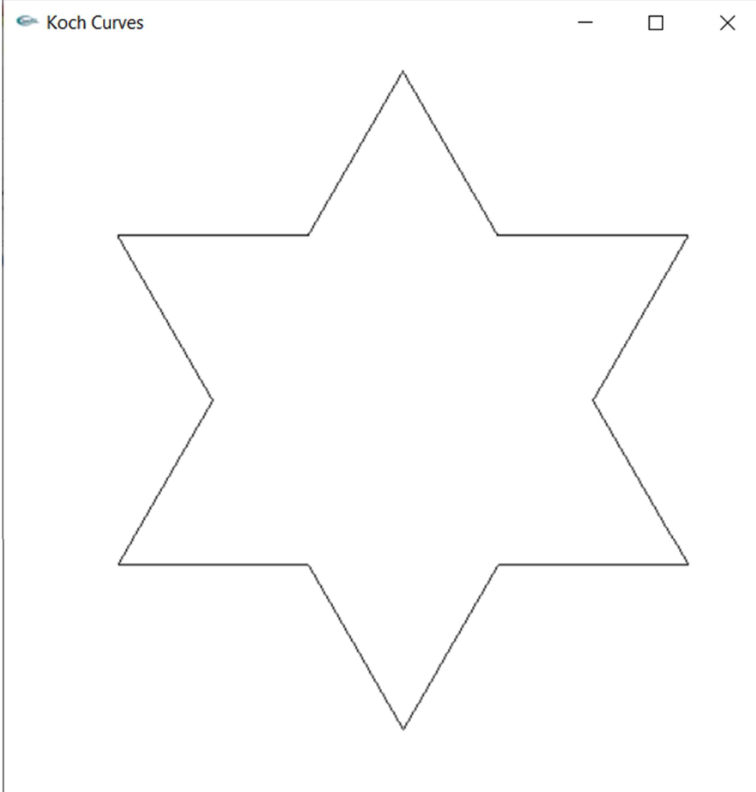
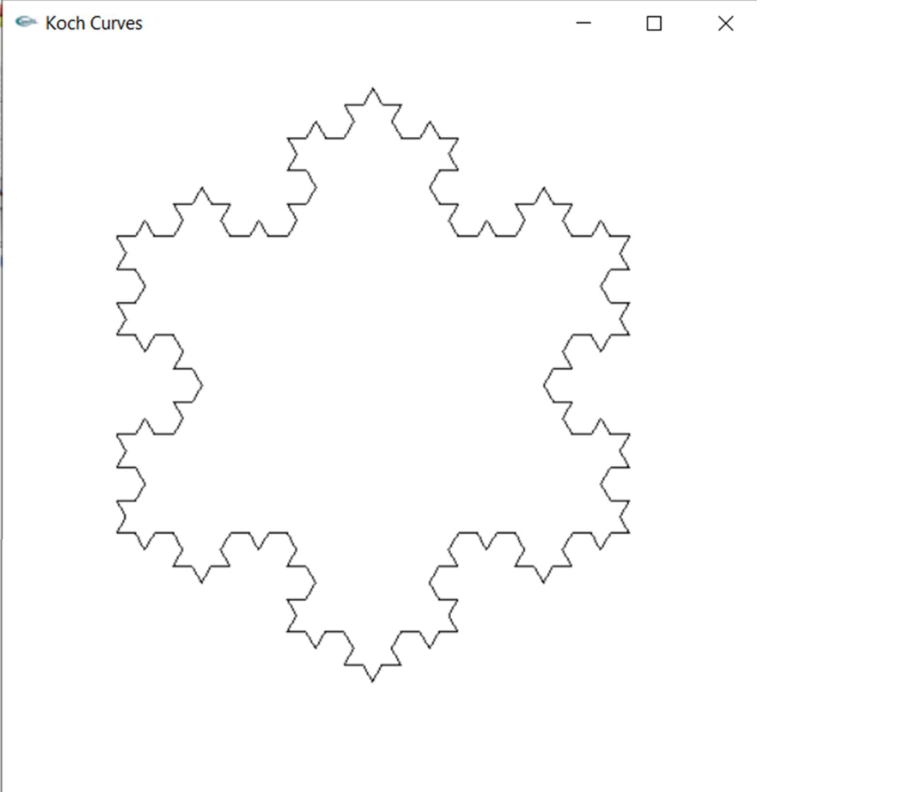
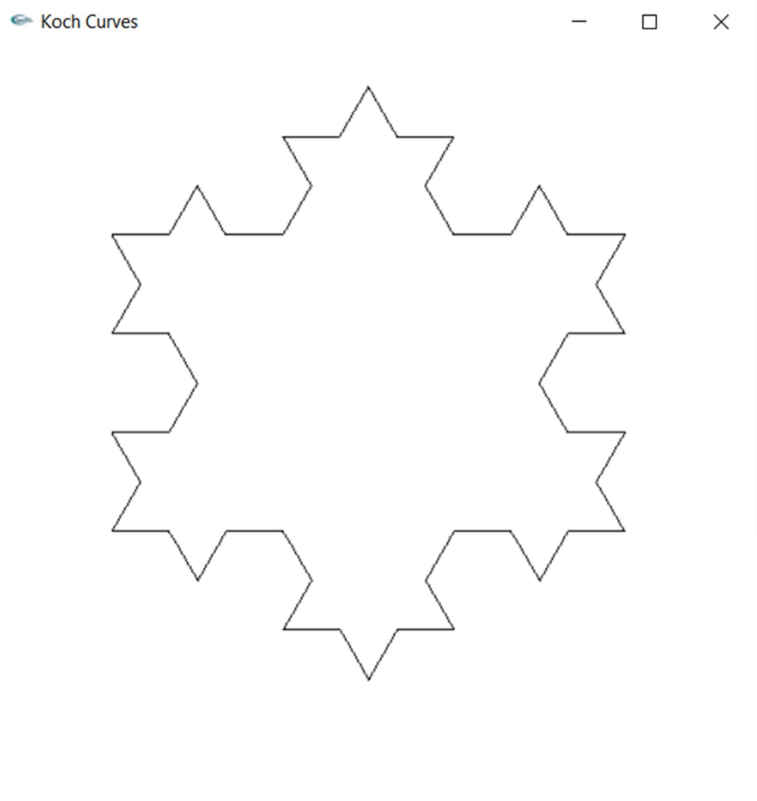
SE I.T

**Generate fractal patterns using i) Bezier ii) Koch Curve.**

INPUT

Input-  
#include<windows.h>  
#include <GL/glut.h>  
#include <math.h>  
GLfloat oldx=-0.7,oldy=0.5;  
void drawkoch(GLfloat dir,GLfloat len,GLint iter)  
{  
Hint: First write a function to calculate any specific point on curve B for any t, given all the  
control points (clicks) set so far.  
Show the TA that your Bézier curve can be rendered with more than 3 control points. An  
example of what multiple calls to your Bézier render function can do is shown below varying  
with the number of line segments and 6 control points:  
GLdouble dirRad = 0.0174533 \* dir ;  
GLfloat newX = oldx + len \* cos(dirRad);  
GLfloat newY = oldy + len \* sin(dirRad);  
if (iter==0)  
{  
glVertex2f(oldx, oldy);  
glVertex2f(newX, newY);  
oldx = newX;  
oldy = newY;  
}  
else  
{  
iter--;  
//draw the four parts of the side \_/\\_  
drawkoch(dir, len, iter);  
dir += 60.0;  
drawkoch(dir, len, iter);  
dir -= 120.0;  
drawkoch(dir, len, iter);  
dir += 60.0;  
drawkoch(dir, len, iter);  
}  
}  
void display()  
{  
glClearColor(1.0,1.0,1.0,0);  
glColor3f(0.0, 0.0, 0.0);  
glClear( GL\_COLOR\_BUFFER\_BIT );  
glBegin(GL\_LINES);  
drawkoch(0.0,0.5,1);  
drawkoch(-120.0, 0.5, 1);  
drawkoch(120.0,0.5,1);  
/\*  
drawkoch(0.0,0.15,2);  
drawkoch(-120.0, 0.15, 2);  
drawkoch(120.0,0.15,2);  
/\*  
drawkoch(0.0,0.05,3);  
drawkoch(-120.0, 0.05, 3);  
drawkoch(120.0,0.05,3);  
\*/  
glEnd();  
glFlush();  
}  
int main(int argc, char\*\* argv)  
{  
glutInit(&argc,argv);  
glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB);  
glutInitWindowSize(500,500);  
glutInitWindowPosition(0,0);  
glutCreateWindow("Koch Curves");  
glutDisplayFunc(display);  
glutMainLoop();  
}

OUTPUT



Name : Dishank Shekokare

Roll no. : 5216

SE I.T

**Implement animation principles for any object.**

INPUT

#include <GL/gl.h>  
#include<windows.h>  
#include <GL/glut.h>  
#include <math.h>  
const double PI = 3.141592654;  
int frameNumber = 0;  
void drawDisk(double radius) {  
int d;  
glBegin(GL\_POLYGON);  
for (d = 0; d < 32; d++) {  
double angle = 2\*PI/32 \* d;  
glVertex2d( radius\*cos(angle), radius\*sin(angle));  
}  
glEnd();  
}  
void drawship() {  
glColor3f(1.5,1,0);  
glBegin(GL\_POLYGON);  
glVertex2f(1.0f,5);//x1  
glVertex2f(-2.0f,4.5);//x2  
glVertex2f(1.0f,4);//y2  
glEnd();  
glColor3f(1.5,0,1);  
glBegin(GL\_LINES);  
glVertex2f(1.0f,2);//x1  
glVertex2f(1.0f,4);//y2  
glEnd();  
glColor3f(1,0,0);  
glBegin(GL\_POLYGON);  
glVertex2f(-2.0f,0);//x1  
glVertex2f(2.0f,0);//x2  
glVertex2f(3.0f,2);//y2  
glVertex2f(-3.0f,2);//y1  
glEnd();  
}  
void drawSun() {  
int i;  
glColor3f(1.0,0.8,0.0);  
for (i = 0; i < 20; i++) {  
glRotatef( 360 / 20, 0, 0, 1 );  
glBegin(GL\_LINES);  
glVertex2f(0, 0);  
glVertex2f(0.75f, 0);  
glEnd();  
}  
drawDisk(0.5);  
glColor3f(0,0,0);  
}  
void display() {  
glClear(GL\_COLOR\_BUFFER\_BIT);  
glLoadIdentity();  
glColor3f(0, 0.6f, 0.2f);  
glBegin(GL\_POLYGON);  
glVertex2f(-3,-1);  
glVertex2f(1.5f,1.65f);  
glVertex2f(5,-1);  
glEnd();  
glBegin(GL\_POLYGON);  
glVertex2f(-3,-1);  
glVertex2f(3,2.1f);  
glVertex2f(7,-1);  
glEnd();  
glBegin(GL\_POLYGON);  
glVertex2f(0,-1);  
glVertex2f(6,1.2f);  
glVertex2f(20,-1);  
glEnd();  
glColor3f(0.2f, 0.2f, 1.0f);  
glBegin(GL\_POLYGON);  
glVertex2f(0,-0.4f);  
glVertex2f(7,-0.4f);  
glVertex2f(7,0.4f);  
glVertex2f(0,0.4f);  
glEnd();  
glPushMatrix();  
glTranslated(1.8,3,0);  
glRotated(-frameNumber\*2.7,0,0,1);  
drawSun();  
glPopMatrix();  
glPushMatrix();  
glTranslated(-3 + 15\*(frameNumber % 300) / 300.0, 0, 0);  
glScaled(0.3,0.3,1);

drawship();  
glPopMatrix();  
glutSwapBuffers();  
}  
void doFrame(int v) {  
frameNumber++;  
glutPostRedisplay();  
glutTimerFunc(30,doFrame,0);  
}  
void init() {  
glClearColor(0.6f, 0.6f, 1.0f, 0);  
glMatrixMode(GL\_PROJECTION);  
glLoadIdentity();  
glOrtho(0, 7, -1, 4, -1, 1);  
glMatrixMode(GL\_MODELVIEW);  
}  
int main(int argc, char\*\* argv) {  
glutInit(&argc, argv);  
glutInitDisplayMode(GLUT\_DOUBLE);  
glutInitWindowSize(700,500);  
glutInitWindowPosition(100,100);  
glutCreateWindow("Animation");  
init();  
glutDisplayFunc(display);  
glutTimerFunc(200,doFrame,0);  
glutMainLoop();  
return 0;  
}

OUTPUT

