LAB SET 5

Write a program to demonstrate the plotting of implicit functions with marching squares technique.

```
#include <GL/glut.h>
#include <stdio.h>
#define X_MAX 1.0
#define Y_MAX 1.0
#define X_MIN -1.0
#define Y_MIN -1.0
#define N_X 50
#define N_Y 50
#define THRESHOLD 0.0

void display()
{
	double f(double,double);
	int cell(double,double,double,double);
	void lines(int,int,int,double,double,double,double);
	double data[N_X][N_Y];
	int i,j,c;
	glClear(GL_COLOR_BUFFER_BIT);
	for(i=0;i<N_X;i++)
		for(j=0;j<N_Y;j++)
		{
			data[i][j]=f(X_MIN+i*(X_MAX-X_MIN)/(N_X-1.0),Y_MIN+j*(Y_MA
			X-Y_MIN)/(N_Y-1.0));
		}
	for(i=0;i<N_X;i++)
		for(j=0;j<N_Y;j++)
		{
			c=cell(data[i][j],data[i+1][j],data[i+1][j+1],
			data[i][j+1]);

			lines(c,i,j,data[i][j],data[i+1][j],data[i+1][j+1],
			data[i][j+1]);
		}
	glFlush();
}

double f(double x,double y)
{
	double a=0.49;
	double b=0.50;
	return((x*x+y*y+a*a)*(x*x+y*y+a*a)-4*a*a*x*x-b*b*b*b);
}

int cell(double a,double b,double c,double d)
{
	int n=0;
	if(a>THRESHOLD) n+=1;
	if(b>THRESHOLD) n+=2;
	if(c>THRESHOLD) n+=4;
	if(d>THRESHOLD) n+=8;
	return n;
}
```

```
void lines(int num,int i,int j,double a,double b,double c,double d)
{
	void draw_one(int,int,int,double,double,double,double);
	void draw_adjacent(int,int,int,double,double,double,double);
	void draw_opposite(int,int,int,double,double,double,double);
	switch(num)
	{
		case 1:
		case 2:
		case 4:
		case 7:
		case 8:
		case 11:
		case 13:
		case 14:	draw_one(num, i,j,a,b,c,d);
				break;
		case 3:
		case 6:
		case 9:
		case 12:	draw_adjacent(num,i,j,a,b,c,d);
				break;
		case 5:
		case 10:	draw_opposite(num, i,j,a,b,c,d);
				break;
		case 0:
		case 15:	break;
	}
}


void draw_one(int num,int i,int j,double a,double b,double c,double d)
{
	double x1,y1, x2, y2;
	double ox,oy;
	double dx,dy;
	dx=(X_MAX-(X_MIN))/(N_X-1.0);
	dy=(Y_MAX-(Y_MIN))/(N_Y-1.0);
	ox=X_MIN+i*(X_MAX-(X_MIN))/(N_X-1.0);
	oy=Y_MIN+j*(Y_MAX-(Y_MIN))/(N_Y-1.0);
	switch(num)
	{
		case 1:
		case 14:	x1=ox;
				y1=oy+dy*(THRESHOLD-a)/(d-a);
				x2=ox+dx*(THRESHOLD-a)/(b-a);
				y2=oy;
				break;
		case 7:
		case 8:		x1=ox;
				y1=oy+dy*(THRESHOLD-a)/(d-a);
				x2=ox+dx*(THRESHOLD-d)/(c-d);
				y2=oy+dy;
				break;
		case 4:
		case 11:	x1=ox+dx*(THRESHOLD-d)/(c-d);
				y1=oy+dy;
				x2=ox+dx;
				y2=oy+dy*(THRESHOLD-b)/(c-b);
```

```
                        break;
                case 2:
                case 13:    x1=ox+dx*(THRESHOLD-a)/(b-a);
                            y1=oy;
                            x2=ox+dx;
                            y2=oy+dy*(THRESHOLD-b)/(c-b);
                            break;
        }
        glBegin(GL_LINES);
        glVertex2d(x1, y1);
        glVertex2d(x2, y2);
        glEnd();
}


void draw_adjacent(int num,int i,int j,double a,double b,double c,double d)
{
        double x1,y1,x2,y2;
        double ox,oy;
        double dx,dy;
        dx=(X_MAX-(X_MIN))/(N_X-1.0);
        dy=(Y_MAX-(Y_MIN))/(N_Y-1.0);
        ox=X_MIN+i*(X_MAX-(X_MIN))/(N_X-1.0);
        oy=Y_MIN+j*(Y_MAX-(Y_MIN))/(N_Y-1.0);
        switch(num)
        {
                case 6:
                case 9:     x1=ox+dx*(THRESHOLD-a)/(b-a);
                            y1=oy;
                            x2=ox+dx*(THRESHOLD-d)/(c-d);
                            y2=oy+dy;
                            break;
                case 3:
                case 12:    x1=ox;
                            y1=oy+dy*(THRESHOLD-a)/(d-a);
                            x2=ox+dx;
                            y2=oy+dy*(THRESHOLD-b)/(c-b);
                            break;
        }
        glBegin(GL_LINES);
        glVertex2d(x1, y1);
        glVertex2d(x2, y2);
        glEnd();
}


void draw_opposite(int num,int i,int j,double a,double b,double c,double d)
{
        double x1,y1,x2,y2,x3,y3,x4,y4;
        double ox,oy;
        double dx,dy;
        dx=(X_MAX-(X_MIN))/(N_X-1.0);
        dy=(Y_MAX-(Y_MIN))/(N_Y-1.0);
        ox=X_MIN+i*(X_MAX-(X_MIN))/(N_X-1.0);
        oy=Y_MIN+j*(Y_MAX-(Y_MIN))/(N_Y-1.0);
```

```
	switch(num)
	{
		case 10:	x1=ox;
				y1=oy+dy*(THRESHOLD-a)/(d-a);
				x2=ox+dx*(THRESHOLD-a)/(b-a);
				y2=oy;
				x3=ox+dx*(THRESHOLD-d)/(c-d);
				y3=oy+dy;
				x4=ox+dx;
				y4=oy+dy*(THRESHOLD-b)/(c-b);
				break;
		case 5:		x1=ox;
				y1=oy+dy*(THRESHOLD-a)/(d-a);
				x2=ox+dx*(THRESHOLD-d)/(c-d);
				y2=oy+dy;
				x3=ox+dy*(THRESHOLD-a)/(b-a);
				y3=oy;
				x4=ox+dx;
				y4=oy+dy*(THRESHOLD-b)/(c-b);
				break;
	}
	glBegin(GL_LINES);
	glVertex2d(x1,y1);
	glVertex2d(x2,y2);
	glVertex2d(x3,y3);
	glVertex2d(x4,y4);
	glEnd();
}



void myReshape(int w,int h)
{
	glViewport(0,0,w,h);
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	if(w<=h)
		gluOrtho2D(X_MIN,X_MAX,Y_MIN*(GLfloat)h/(GLfloat)w,
		Y_MAX*(GLfloat)h/(GLfloat)w);
	else
		gluOrtho2D(X_MIN*(GLfloat)w/(GLfloat)h,
		X_MAX*(GLfloat)w/(GLfloat)h,Y_MIN,Y_MAX);

	glMatrixMode(GL_MODELVIEW);
}

int main(int argc, char **argv)
{
	glutInit(&argc, argv);
	glutInitWindowSize(500, 500);
	glutCreateWindow("Akarsh Singh//1SI16CS007//LabSet 5");
	glutReshapeFunc(myReshape);
	glutDisplayFunc(display);
	glClearColor(0.0,0.0,0.0,1.0);
	glColor3f(1.0,1.0,1.0);
	glutMainLoop();
	return 0;
}
```

OUTPUT