Program-1

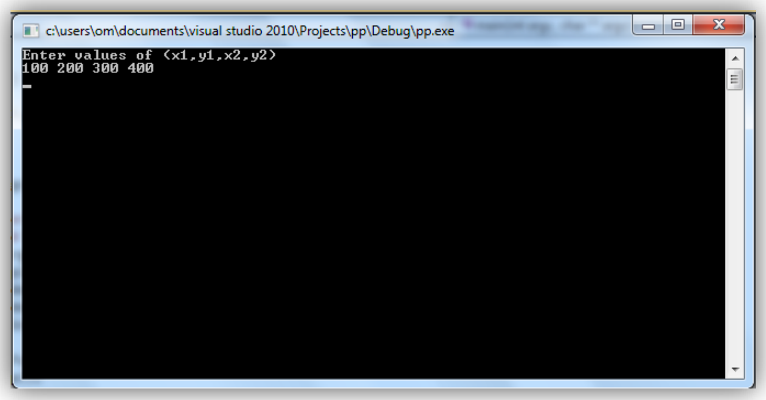
Implement Brenham's line drawing algorithm for all types of slope

```
#include "stdafx.h"
#include<stdio.h>
#include<stdlib.h>
#include<GL/glut.h>

int x1,y1,x2,y2;

void myInit()
{
	glClear(GL_COLOR_BUFFER_BIT);
	glClearColor(0.0,0.0,0.0,1.0);
	glMatrixMode(GL_PROJECTION);
	gluOrtho2D(0,500,0,500);
}

void draw_pixel(int x,int y)
{
	glBegin(GL_POINTS);
	glVertex2i(x,y);
	glEnd();
}

void draw_line(int x1,int x2,int y1,int y2)
{
	int dx,dy,i,e;
	int incx,incy,inc1,inc2;
	int x,y;
	dx=x2-x1;
	dy=y2-y1;
	if(dx<0) dx=-dx;
	if(dy<0) dy=-dy;
	incx=1;
	if(x2<x1) incx=-1;
	incy=1;
	if(y2<y1) incy=-1;
	x=x1;
	y=y1;
	if(dx>dy)
	{
		draw_pixel(x,y);
		e=2*dy-dx;
		inc1=2*(dy-dx);
		inc2=2*dy;
```

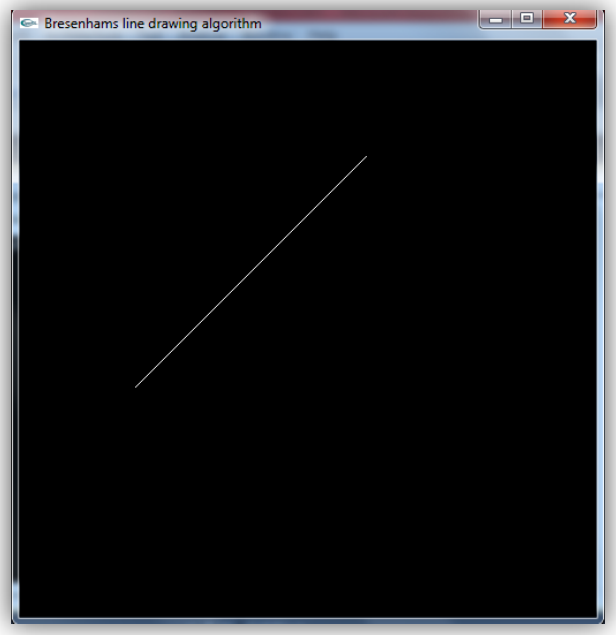
```
        for(i=0;i<dx;i++)
        {
            if(e>=0)
            {
                y+=incy;
                e+=inc1;
            }
            else
                e+=inc2;
             x+=incx;
            draw_pixel(x,y);
        }
    }
    else
    {
        draw_pixel(x,y);
        e=2*dx-dy;
        inc1=2*(dx-dy);
        inc2=2*dx;
        for(i=0;i<dy;i++)
        {
            if(e>=0)
            {
                x+=incx;
                e+=inc1;
            }
            else
                e+=inc2;
                y+=incy;
            draw_pixel(x,y);
        }
    }
}

void myDisplay()
{
    draw_line(x1,x2,y1,y2);
    glFlush();
}

int main(int argc,char **argv)
{
    printf("Enter values of (x1,y1,x2,y2)\n");
    scanf("%d %d %d %d",&x1,&y1,&x2,&y2);
    glutInit(&argc,argv);
    glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
    glutInitWindowSize(500,500);
    glutInitWindowPosition(0,0);
```

```
        glutCreateWindow("Bresenhams line drawing algorithm");
        myInit();
        glutDisplayFunc(myDisplay);
        glutMainLoop();
        return 0;
}
```

Output:


c:\users\om\documents\visual studio 2010\Projects\pp\Debug\pp.exe

```
Enter values of (x1,y1,x2,y2)
100 200 300 400
_
```



Program-2

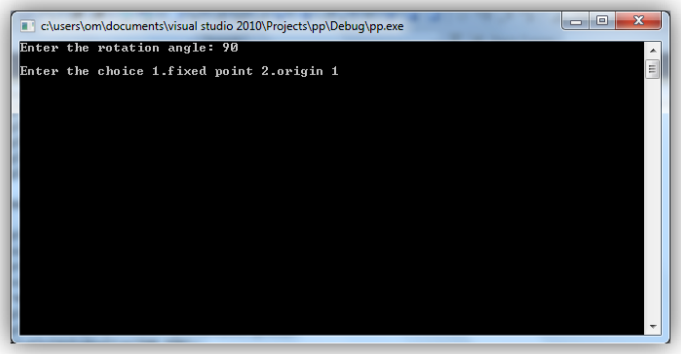
Create and rotate a triangle about the origin and a fixed point

```
#include "stdafx.h"
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<GL/glut.h>

GLfloat house[3][3]={{100.0,150.0,200.0},{100.0,150.0,100.0},{1.0,1.0,1.0}};
GLfloat rot_mat[3][3]={{0},{0},{0}};
GLfloat result[3][3]={{0},{0},{0}};
GLfloat h;
GLfloat k;
GLfloat theta,rad;
int ch;

void multiply()
{
    int i,j,l;
    for(i=0;i<3;i++)
        for(j=0;j<3;j++)
        {
            result[i][j]=0;
            for(l=0;l<3;l++)
                result[i][j]=result[i][j]+rot_mat[i][l]*house[l][j];
        }
}

void rotate()
{
    GLfloat m,n;
    m=-h*(cos(theta)-1)+k*(sin(theta));
    n=-k*(cos(theta)-1)-h*(sin(theta));
    rot_mat[0][0]=cos(theta);
    rot_mat[0][1]=-sin(theta);
    rot_mat[0][2]=m;
    rot_mat[1][0]=sin(theta);
    rot_mat[1][1]=cos(theta);
    rot_mat[1][2]=n;
    rot_mat[2][0]=0;
    rot_mat[2][1]=0;
    rot_mat[2][2]=1;
    multiply();
}
```

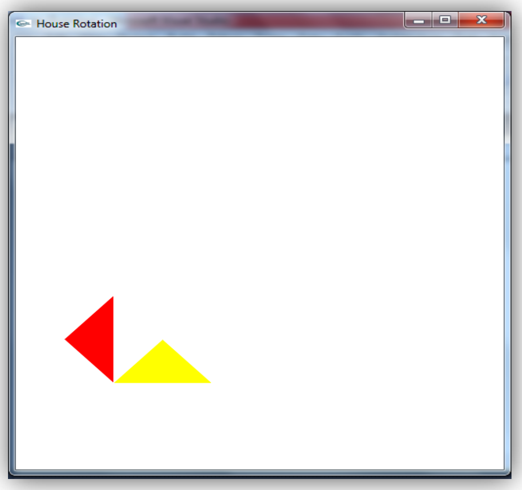
```
void drawhouse(GLfloat mat[3][3])
{
    glBegin(GL_TRIANGLES);
    glVertex2f(mat[0][0],mat[1][0]);
    glVertex2f(mat[0][1],mat[1][1]);
    glVertex2f(mat[0][2],mat[1][2]);
    glEnd();
}

void display()
{
    glClear(GL_COLOR_BUFFER_BIT);
    theta=rad;
    glColor3f(1.0,1.0,0.0);
    drawhouse(house);
    if(ch==1)
    {
        h=100;
        k=100;
        rotate();
        glColor3f(1.0,0.0,0.0);
    }
    if(ch==2)
    {
        h=(house[0][0]+house[0][1]+house[0][2])/3;
        k=(house[1][0]+house[1][1]+house[1][2])/3;
        rotate();
        glColor3f(1.0,0.0,1.0);
    }
    drawhouse(result);
    glFlush();
}

void myinit()
{
    glClearColor(1.0,1.0,1.0,1.0);
    glColor3f(1.0,0.0,0.0);
    glPointSize(1.0);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0.0,499.0,0.0,499.0);
}
```

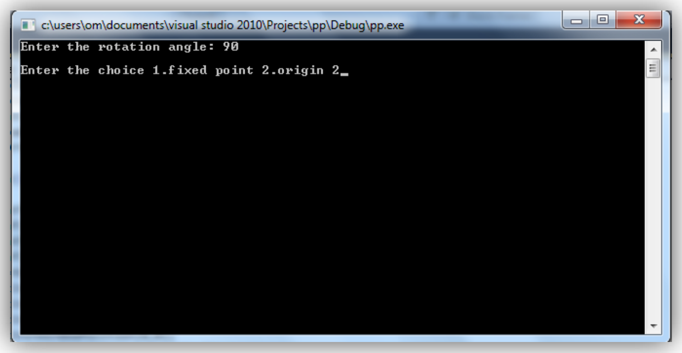
```
int main(int argc,char**argv)
{
    printf("Enter the rotation angle:");
    scanf("%f",&theta);
    printf("\nEnter the choice 1.fixed point 2.origin ");
    scanf("%d",&ch);
    rad=theta*(3.14/180.0);
    glutInit(&argc,argv);
    glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
    glutInitWindowSize(500,500);
    glutInitWindowPosition(0,0);
    glutCreateWindow("House Rotation");
    glutDisplayFunc(display);
    myinit();
    glutMainLoop();
}
```

Output:

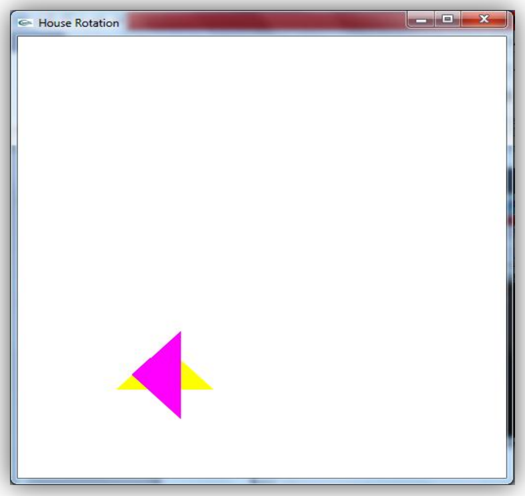

c:\users\om\documents\visual studio 2010\Projects\pp\Debug\pp.exe

```
Enter the rotation angle: 90
Enter the choice 1.fixed point 2.origin 1
```




c:\users\om\documents\visual studio 2010\Projects\pp\Debug\pp.exe

```
Enter the rotation angle: 90
Enter the choice 1.fixed point 2.origin 2_
```



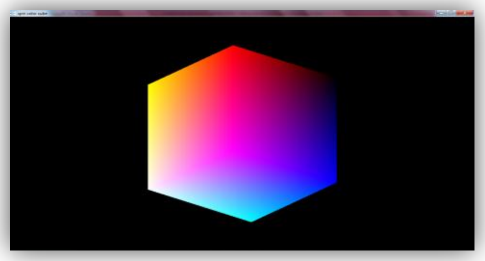
Program-3

To draw a color cube and spin it using OpenGL transformation matrices

```
#include "stdafx.h"
#include<gl/glut.h>
#include<stdio.h>

GLfloat vertices[][3]={{-1.0,-1.0,1.0},{-1.0,1.0,1.0},{1.0,1.0,1.0},{1.0,-1.0,1.0},
                       {-1.0,-1.0,-1.0},{-1.0,1.0,-1.0},{1.0,1.0,-1.0},{1.0,-1.0,-1.0}};
GLfloat colors[][3]={{0.0,0.0,0.0},{1.0,0.0,0.0},{1.0,1.0,0.0},{0.0,1.0,0.0},
                     {0.0,0.0,1.0},{1.0,0.0,1.0},{1.0,1.0,1.0},{0.0,1.0,1.0}};
static GLfloat theta[]={0.0,0.0,0.0};
static GLint axis=2;

void polygon(int a,int b,int c,int d)
{
        glBegin(GL_POLYGON);
        glColor3fv(colors[a]);
        glVertex3fv(vertices[a]);
        glColor3fv(colors[b]);
        glVertex3fv(vertices[b]);
        glColor3fv(colors[c]);
        glVertex3fv(vertices[c]);
        glColor3fv(colors[d]);
        glVertex3fv(vertices[d]);
        glEnd();
}

void colorcube()
{
        polygon(0,3,2,1);
        polygon(2,3,7,6);
        polygon(0,4,7,3);
        polygon(1,2,6,5);
        polygon(4,5,6,7);
        polygon(0,1,5,4);
}

void display()
{
        glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
        glLoadIdentity();
        glRotatef(theta[0],1.0,0.0,0.0);
        glRotatef(theta[1],0.0,1.0,0.0);
        glRotatef(theta[2],0.0,0.0,1.0);
        colorcube();
        glFlush();
        glutSwapBuffers();
}
```

```
void spincube()
{
        theta[axis]+=0.5;
        if(theta[axis]>360.0)theta[axis]-=360.0;
        glutPostRedisplay();
}

void mouse(int btn,int state,int x,int y)
{
        if(btn==GLUT_LEFT_BUTTON && state==GLUT_DOWN) axis=0;
        if(btn==GLUT_MIDDLE_BUTTON && state==GLUT_DOWN) axis=1;
        if(btn==GLUT_RIGHT_BUTTON && state==GLUT_DOWN) axis=2;
}

void myReshape(int w,int h)
{
         glViewport(0,0,w,h);
         glMatrixMode(GL_PROJECTION);
         glLoadIdentity();
         if(w<=h)
                 glOrtho(-2.0,2.0,-2.0*(GLfloat)h/(GLfloat)w,2.0*(GLfloat)h/(GLfloat)w,-10.0,10.0);
         else
                 glOrtho(-2.0*(GLfloat)w/(GLfloat)h,2.0*(GLfloat)w/(GLfloat)h,-2.0,2.0,-10.0,10.0);
        glMatrixMode(GL_MODELVIEW);
}
void main(int argc,char** argv)
{
         glutInit(&argc,argv);
         glutInitDisplayMode(GLUT_DOUBLE|GLUT_RGB|GLUT_DEPTH);
         glutInitWindowSize(500,500);
         glutCreateWindow("spin color cube");
         glutReshapeFunc(myReshape);
         glutDisplayFunc(display);
         glutIdleFunc(spincube);
         glutMouseFunc(mouse);
         glEnable(GL_DEPTH_TEST);
         glutMainLoop();
}
```

Output:



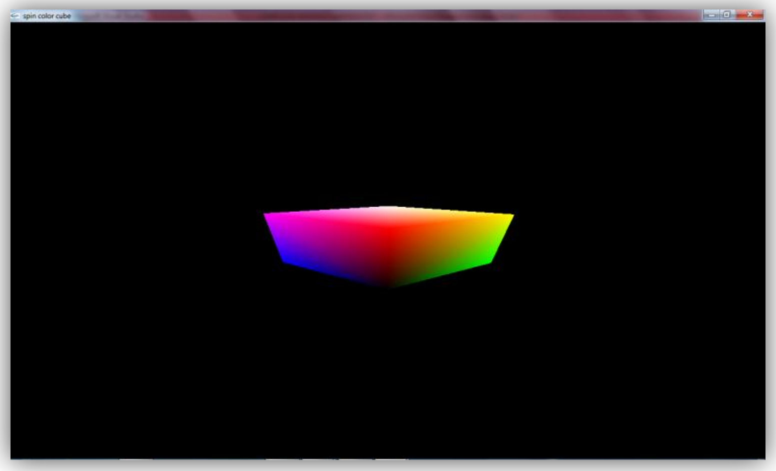
Program-4

To draw a color cube and allow the user to move the camera suitably to experiment with perspective viewing.

```
#include "stdafx.h"
#include<gl/glut.h>
#include<stdio.h>

GLfloat vertices[][3]={{-1.0,-1.0,1.0},{-1.0,1.0,1.0},{1.0,1.0,1.0},{1.0,-1.0,1.0},
                        {-1.0,-1.0,-1.0},{-1.0,1.0,-1.0},{1.0,1.0,-1.0},{1.0,-1.0,-1.0}};
GLfloat colors[][3]={{0.0,0.0,0.0},{1.0,0.0,0.0},{1.0,1.0,0.0},{0.0,1.0,0.0},
                        {0.0,0.0,1.0},{1.0,0.0,1.0},{1.0,1.0,1.0},{0.0,1.0,1.0}};
static GLfloat theta[]={0.0,0.0,0.0};
static GLint axis=2;
static GLfloat viwer[]={0.0,0.0,5.0};

void polygon(int a,int b,int c,int d)
{
        glBegin(GL_POLYGON);
        glColor3fv(colors[a]);
        glVertex3fv(vertices[a]);
        glColor3fv(colors[b]);
        glVertex3fv(vertices[b]);
        glColor3fv(colors[c]);
        glVertex3fv(vertices[c]);
        glColor3fv(colors[d]);
        glVertex3fv(vertices[d]);
        glEnd();
}

void colorcube()
{
        polygon(0,3,2,1);
        polygon(2,3,7,6);
        polygon(0,4,7,3);
        polygon(1,2,6,5);
        polygon(4,5,6,7);
        polygon(0,1,5,4);
}

void display()
{
        glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
        glLoadIdentity();
        gluLookAt(viwer[0],viwer[1],viwer[2],0.0,0.0,0.0,0.0,1.0,0.0);
        glRotatef(theta[0],1.0,0.0,0.0);
        glRotatef(theta[1],0.0,1.0,0.0);
        glRotatef(theta[2],0.0,0.0,1.0);
        colorcube();
        glFlush();
        glutSwapBuffers();
}
```

```
void mouse(int btn,int state,int x,int y)
{
        if(btn==GLUT_LEFT_BUTTON && state==GLUT_DOWN) axis=0;
        if(btn==GLUT_MIDDLE_BUTTON && state==GLUT_DOWN) axis=1;
        if(btn==GLUT_RIGHT_BUTTON && state==GLUT_DOWN) axis=2;
        theta[axis]+=0.5;
        if(theta[axis]>360.0)theta[axis]-=360.0;
        display();
}

void keys(unsigned char key,int x,int y)
{
        if(key=='x') viwer[0]-=1.0;
        if(key=='X') viwer[0]+=1.0;
        if(key=='y') viwer[1]-=1.0;
        if(key=='Y') viwer[1]+=1.0;
        if(key=='z') viwer[2]-=1.0;
        if(key=='Z') viwer[2]+=1.0;
        display();
}

void myReshape(int w,int h)
{
        glViewport(0,0,w,h);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        if(w<=h)
                glFrustum(-2.0,2.0,-2.0*(GLfloat)h/(GLfloat)w,2.0*(GLfloat)h/(GLfloat)w,2.0,0.0);
        else
                glFrustum(-2.0,2.0,-2.0*(GLfloat)w/(GLfloat)h,2.0*(GLfloat)w/(GLfloat)h,2.0,20.0);
        glMatrixMode(GL_MODELVIEW);
}

void main(int argc,char** argv)
{
        glutInit(&argc,argv);
        glutInitDisplayMode(GLUT_DOUBLE|GLUT_RGB|GLUT_DEPTH);
        glutInitWindowSize(500,500);
        glutCreateWindow("spin color cube");
        glutReshapeFunc(myReshape);
        glutDisplayFunc(display);
        glutMouseFunc(mouse);
        glutKeyboardFunc(keys);
        glEnable(GL_DEPTH_TEST);
        glutMainLoop();
}
```

Output:



Program-5

Clip a lines using Cohen-Sutherland algorithm

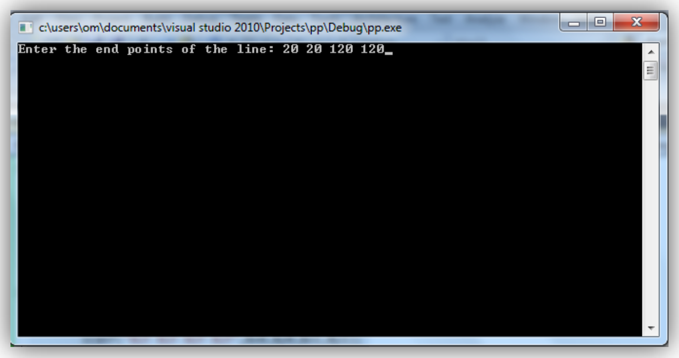
```
#include "stdafx.h"
#include<stdio.h>
#include<GL/glut.h>
#define outcode int

double xmin=50,ymin=50,xmax=100,ymax=100;
double xvmin=200,yvmin=200,xvmax=300,yvmax=300;
double x0,y0,x1,y1;
const int RIGHT=8;
const int LEFT=2;
const int TOP=4;
const int BOTTOM=1;

outcode coc(double x,double y);

void cs(double x0,double y0,double x1,double y1)
{
        outcode oc0,oc1,oco;
        bool accept=false,done=false;
        oc0=coc(x0,y0);
        oc1=coc(x1,y1);
        do
        {
                if(!(oc0|oc1))
                {
                        accept=true;
                        done=true;
                }
                else if(oc0&oc1)
                        done=true;
                else
                {
                        double x,y;
                        oco=oc0?oc0:oc1;
                        if(oco&TOP)
                        {
                                x=x0+(x1-x0)*(ymax-y0)/(y1-y0);
                                y=ymax;
                        }
                        else if(oco&BOTTOM)
                        {
                                x=x0+(x1-x0)*(ymin-y0)/(y1-y0);
                                y=ymin;
                        }
```

```
            else if(oco&RIGHT)
            {
                y=y0+(y1-y0)*(xmax-x0)/(x1-x0);
                x=xmax;
            }
            else
            {
                y=y0+(y1-y0)*(xmin-x0)/(x1-x0);
                x=xmin;
            }
            if(oco==oc0)
            {
                x0=x;
                y0=y;
                oc0=coc(x0,y0);
            }
            else
            {
                x1=x;
                y1=y;
                oc1=coc(x1,y1);
            }
        }
    }
    while(!done);
    if(accept)
    {
        double sx=(xvmax-xvmin)/(xmax-xmin);
        double sy=(yvmax-yvmin)/(ymax-ymin);
        double vx0=xvmin+(x0-xmin)*sx;
        double vy0=yvmin+(y0-ymin)*sy;
        double vx1=xvmin+(x1-xmin)*sx;
        double vy1=yvmin+(y1-ymin)*sy;
        glColor3f(1.0,0.0,0.0);
        glBegin(GL_LINE_LOOP);
        glVertex2f(xvmin,yvmin);
        glVertex2f(xvmax,yvmin);
        glVertex2f(xvmax,yvmax);
        glVertex2f(xvmin,yvmax);
        glEnd();
        glColor3f(0.0,0.0,1.0);
        glBegin(GL_LINES);
        glVertex2d(vx0,vy0);
        glVertex2d(vx1,vy1);
        glEnd();
    }
}
```

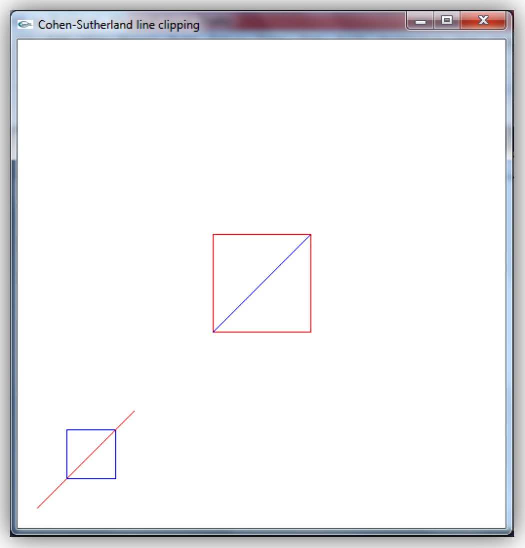
```
outcode coc(double x,double y)
{
        outcode c=0;
        if(y>ymax)
                c=TOP;
        else if(y<ymin)
                c=BOTTOM;
        if(x>xmax)
                c=RIGHT;
        else if(x<xmin)
                c=LEFT;
        return c;
}

void display()
{
        glClear(GL_COLOR_BUFFER_BIT);
        glColor3f(1.0,0.0,0.0);
        glBegin(GL_LINES);
        glVertex2d(x0,y0);
        glVertex2d(x1,y1);
        glEnd();
        glColor3f(0.0,0.0,1.0);
        glBegin(GL_LINE_LOOP);
        glVertex2f(xmin,ymin);
        glVertex2f(xmax,ymin);
        glVertex2f(xmax,ymax);
        glVertex2f(xmin,ymax);
        glEnd();
        cs(x0,y0,x1,y1);
        glFlush();
}

void myinit()
{
        glClearColor(1.0,1.0,1.0,1.0);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        gluOrtho2D(0.0,499.0,0.0,499.0);
}

void main(int argc,char ** argv)
{
                printf("Enter the end points of the line:");
                scanf("%lf %lf %lf %lf",&x0,&y0,&x1,&y1);
                glutInit(&argc,argv);
                glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
                glutInitWindowSize(500,500);
                glutInitWindowPosition(0,0);
```

```
            glutCreateWindow("Cohen-Sutherland line clipping");
            glutDisplayFunc(display);
            myinit();
            glutMainLoop();
}
```

Output:


c:\users\om\documents\visual studio 2010\Projects\pp\Debug\pp.exe

```
Enter the end points of the line: 20 20 120 120_
```

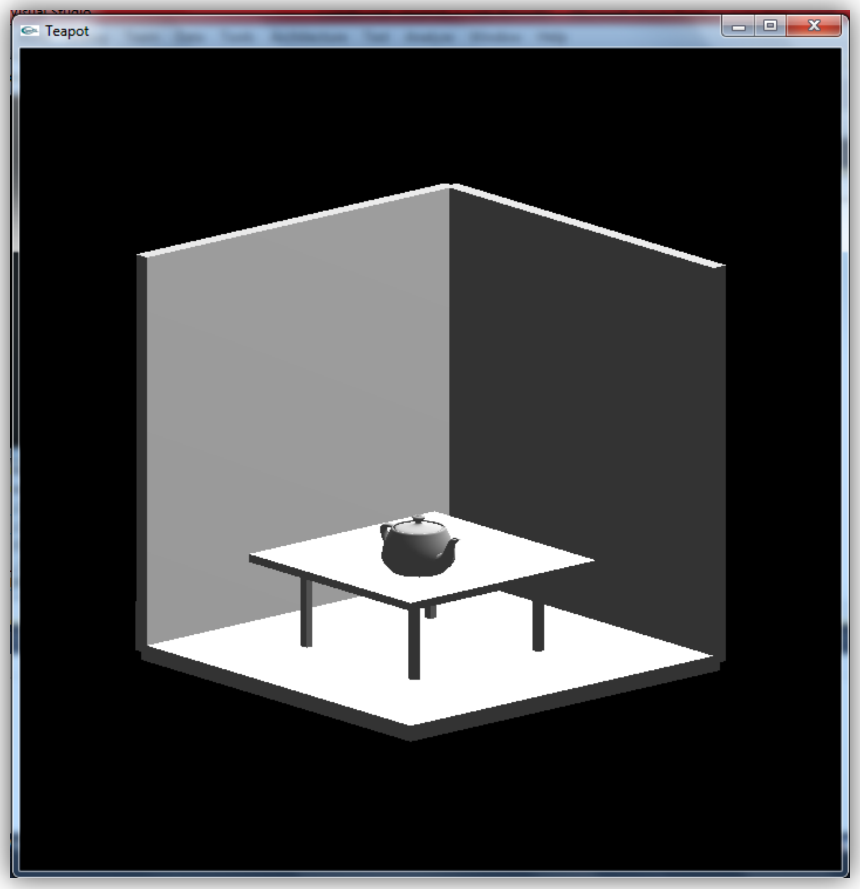


Program-6

To draw a simple shaded scene consisting of a tea pot on a table. Define suitably the position and properties of light source along with the properties of the surfaces of the solid object used in the scene.

```
#include "stdafx.h"
#include<gl/glut.h>

void obj(double tx,double ty,double tz,double sx,double sy,double sz)
{
        glRotated(50,0,1,0);
        glRotated(10,-1,0,0);
        glRotated(11.7,0,0,-1);
        glTranslated(tx,ty,tz);
        glScaled(sx,sy,sz);
        glutSolidCube(1);
        glLoadIdentity();
}

void display()
{
        glViewport(0,0,700,700);
        glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
        obj(0,0,0.5,1,1,0.04);
        obj(0,-0.5,0,1,0.04,1);
        obj(-0.5,0,0,0.04,1,1);
        obj(0,-0.3,0,0.02,0.2,0.02);
        obj(0,-0.3,-0.4,0.02,0.2,0.02);
        obj(0.4,-0.3,0,0.02,0.2,0.02);
        obj(0.4,-0.3,-0.4,0.02,0.2,0.02);
        obj(0.2,-0.18,-0.2,0.6,0.02,0.6);
        glRotated(50,0,1,0);
        glRotated(10,-1,0,0);
        glRotated(11.7,0,0,-1);
        glTranslated(0.3,-0.1,-0.3);
        glutSolidTeapot(0.09);
        glFlush();
        glLoadIdentity();
}

void main()
{
        float ambient[]={1,1,1,1};
        float light_pos[]={27,80,2,3};
        glutInitWindowSize(700,700);
        glutCreateWindow("Teapot");
        glutDisplayFunc(display);
        glEnable(GL_LIGHTING);
        glEnable(GL_LIGHT0);
        glMaterialfv(GL_FRONT,GL_AMBIENT,ambient);
        glLightfv(GL_LIGHT0,GL_POSITION,light_pos);
        glEnable(GL_DEPTH_TEST);
        glutMainLoop();
}
```

Output:



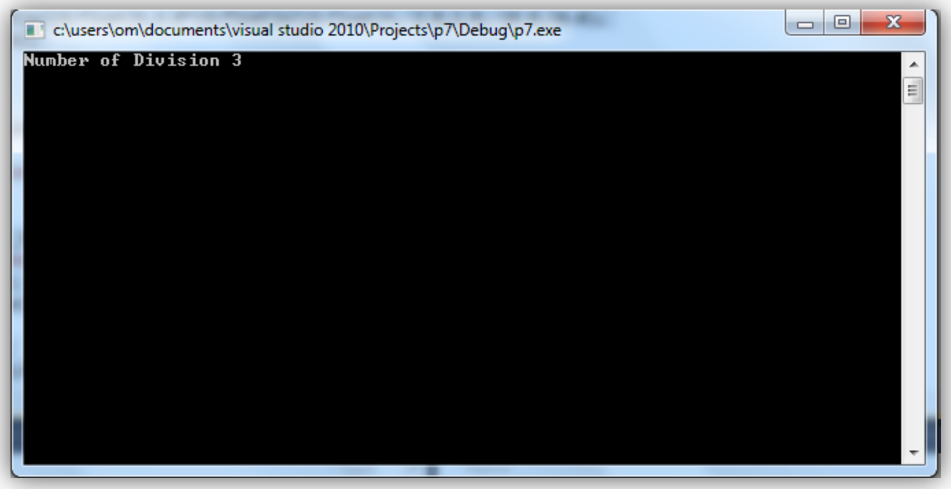
Program-7

To recursively subdivided a Tetrahedron to form 3D Sierpinski Gasket. The number of recursive steps is to be specified by the user

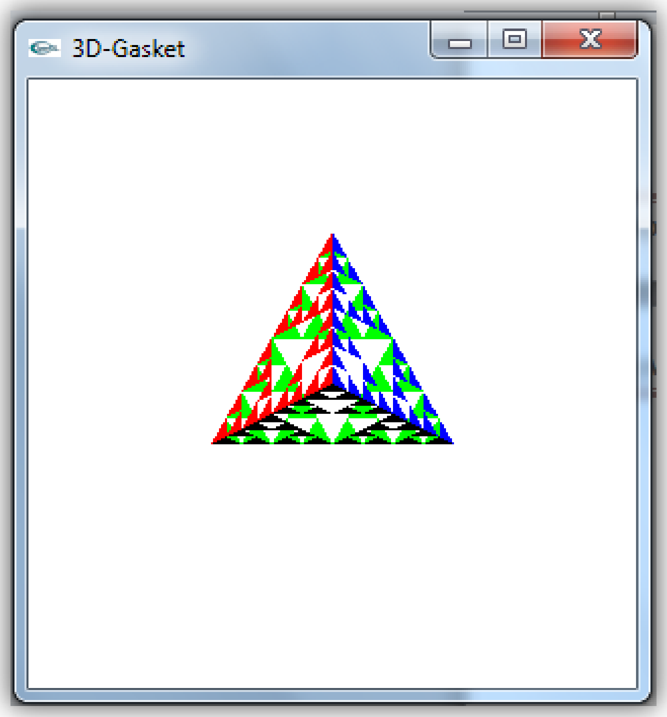
```
#include "stdafx.h"
#include<stdio.h>
#include<gl/glut.h>

typedef float point[3];
point v[]={{0.0,0.0,1.0},{0.0,1.0,-1.0},{-0.8,-0.4,-0.1},{0.8,-0.4,-0.1}};
int n;

void triangle(point a,point b,point c)
{
	glBegin(GL_POLYGON);
		glVertex3fv(a);
		glVertex3fv(b);
		glVertex3fv(c);
	glEnd();
}

void divide_triangle(point a,point b,point c,int m)
{
	point v1,v2,v3;
	int j;
	if(m>0)
	{
		for(j=0;j<3;j++)
			v1[j]=(a[j]+b[j])/2;
		for(j=0;j<3;j++)
			v2[j]=(a[j]+c[j])/2;
		for(j=0;j<3;j++)
			v3[j]=(b[j]+c[j])/2;
		divide_triangle(a,v1,v2,m-1);
		divide_triangle(c,v2,v3,m-1);
		divide_triangle(b,v3,v1,m-1);
	}
	else(triangle(a,b,c));
}
```

```
void tetrahedron(int m)
{
        glColor3f(1.0,0.0,0.0);
        divide_triangle(v[0],v[1],v[2],m);
        glColor3f(0.0,1.0,0.0);
        divide_triangle(v[3],v[2],v[1],m);
        glColor3f(0.0,0.0,1.0);
        divide_triangle(v[0],v[3],v[1],m);
        glColor3f(0.0,0.0,0.0);
        divide_triangle(v[0],v[2],v[3],m);
}

void display(void)
{
        glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
        glLoadIdentity();
        tetrahedron(n);
        glFlush();
}

void myReshape(int w,int h)
{
        glViewport(0,0,w,h);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        if(w<=h)
                glOrtho(-2.0,2.0,-2.0*(GLfloat)h/(GLfloat)w,2.0*(GLfloat)h/(GLfloat)w,-10.0,10.0);
        else
                glOrtho(-2.0*(GLfloat)w/(GLfloat)h,2.0*(GLfloat)w/(GLfloat)h,-2.0,2.0,-10.0,10.0);
        glMatrixMode(GL_MODELVIEW);
        glutPostRedisplay();
}

void main(int argc,char **argv)
{
        printf("Number of Division");
        scanf("%d",&n);
        glutInit(&argc,argv);
        glutInitDisplayMode(GLUT_RGB|GLUT_SINGLE|GLUT_DEPTH);
        glutCreateWindow("3D-Gasket");
        glutDisplayFunc(display);
        glutReshapeFunc(myReshape);
        glEnable(GL_DEPTH_TEST);
        glClearColor(1.0,1.0,1.0,0.0);
        glutMainLoop();
}
```

Output:


c:\users\om\documents\visual studio 2010\Projects\p7\Debug\p7.exe

```
Number of Division 3
```



Program-8

Develop a menu driven program to animate a flag using Bezier curve algorithm

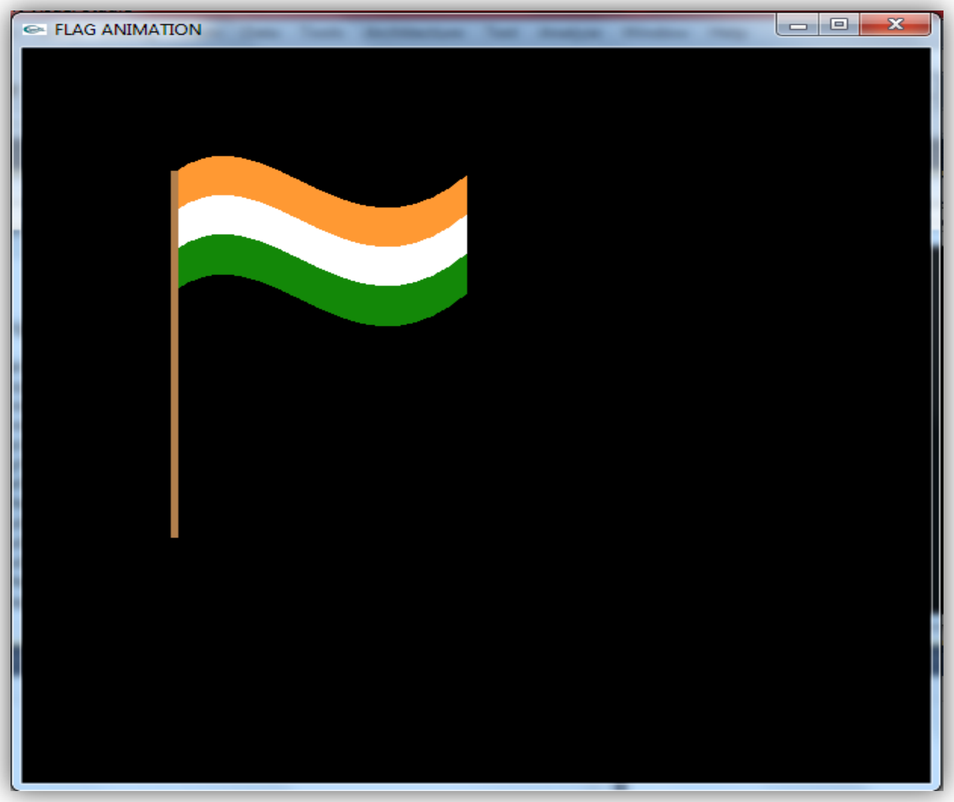
```
#include "stdafx.h"
#include<gl/glut.h>
#include<math.h>
#include<stdlib.h>
#include<stdio.h>

#define PI 3.14
#define WAVE 1
#define STOP 2
#define QUIT 3

GLsizei winWd=600,winHt=600;
GLfloat xmin=0,xmax=120,ymin=0,ymax=120;

typedef struct
{
        GLfloat x,y,z;
}w3d;

void bino(GLint n,GLint *C)
{
        GLint j,k;
        for(k=0;k<=n;k++)
        {
                C[k]=1;
                for(j=n;j>=k+1;j--)
                        C[k]*=j;
                for(j=n-k;j>=2;j--)
                        C[k]/=j;
        }
}

void computept(GLfloat u,w3d *bezpt,GLint ncp,w3d *cp,GLint *C)
{
        GLint k,n=ncp-1;
        GLfloat bezblendfun;
        bezpt->x=bezpt->y=bezpt->z=0.0;
        for(k=0;k<ncp;k++)
        {
                bezblendfun=C[k]*pow(u,k)*pow(1-u,n-k);
                bezpt->x+=cp[k].x*bezblendfun;
                bezpt->y+=cp[k].y*bezblendfun;
                bezpt->z+=cp[k].z*bezblendfun;
        }
}
```

```
void beizer(w3d *cp,GLint ncp,GLint nbc)
{
        w3d bp;
        GLfloat u;
        GLint k,*C;
        C=new GLint[ncp];
        bino(ncp-1,C);
        glBegin(GL_LINE_STRIP);
        for(k=0;k<=nbc;k++)
        {
                u=GLfloat(k)/GLfloat(nbc);
                computept(u,&bp,ncp,cp,C);
                glVertex2f(bp.x,bp.y);
        }
        glEnd();
        delete []C;
}

static float theta=0;

void display()
{
        GLint ncp=4,nbp=20;
        w3d cp[4]={{20,100,0},{30,110,0},{50,90,0},{60,100,0}};
        cp[1].x+=10*sin(theta*PI/180);
        cp[1].y+=5*sin(theta*PI/180);
        cp[2].x-=10*sin((theta+30)*PI/180);
        cp[2].y-=10*sin((theta+30)*PI/180);
        cp[3].x-=4*sin(theta*PI/180);
        cp[3].y+=sin((theta-30)*PI/180);
        glClear(GL_COLOR_BUFFER_BIT);
        glColor3f(1.0,1.0,1.0);
        glPointSize(5);
        glPushMatrix();
        glLineWidth(5);
        glColor3f(255/255.0,153/255.0,51/255.0);
        for(int i=0;i<8;i++)
        {
                glTranslatef(0,-0.8,0);
                beizer(cp,ncp,nbp);
        }
        glColor3f(1,1,1);
        for(int i=0;i<8;i++)
        {
                glTranslatef(0,-0.8,0);
                beizer(cp,ncp,nbp);
        }
```

```
    glColor3f(19/255.0,136/255.0,8/255.0);
    for(int i=0;i<8;i++)
    {
        glTranslatef(0,-0.8,0);
        beizer(cp,ncp,nbp);
    }
    glPopMatrix();
    glColor3f(0.7,0.5,0.3);
    glLineWidth(5);
    glBegin(GL_LINES);
    glVertex2f(20,40);
    glVertex2f(20,100);
    glEnd();
    glFlush();
    glutPostRedisplay();
    glutSwapBuffers();
}

void myReshape(int w,int h)
{
    glViewport(0,0,w,h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(xmin,xmax,ymin,ymax);
    glMatrixMode(GL_MODELVIEW);
    glutPostRedisplay();
}

void animate()
{
    theta+=0.5;
    glutPostRedisplay();
}

void menu(int id)
{
    switch(id)
    {
        case WAVE: glutIdleFunc(animate);
                    break;
        case STOP: glutIdleFunc(NULL);
                    break;
        case QUIT: exit(0);
                    break;
    }
}
```

```
int main(int argc,char ** argv)
{
	glutInit(&argc,argv);
	glutInitDisplayMode(GLUT_RGB|GLUT_DOUBLE);
	glutInitWindowPosition(0,0);
	glutInitWindowSize(winWd,winHt);
	glutCreateWindow("FLAG ANIMATION");
	glutDisplayFunc(display);
	glutReshapeFunc(myReshape);
	glutCreateMenu(menu);
	glutAddMenuEntry("Flag Waving",WAVE);
	glutAddMenuEntry("Stop Waving",STOP);
	glutAddMenuEntry("Quit",QUIT);
	glutAttachMenu(GLUT_RIGHT_BUTTON);
	glutMainLoop();
	return 0;
}
```

Output:



Program-9

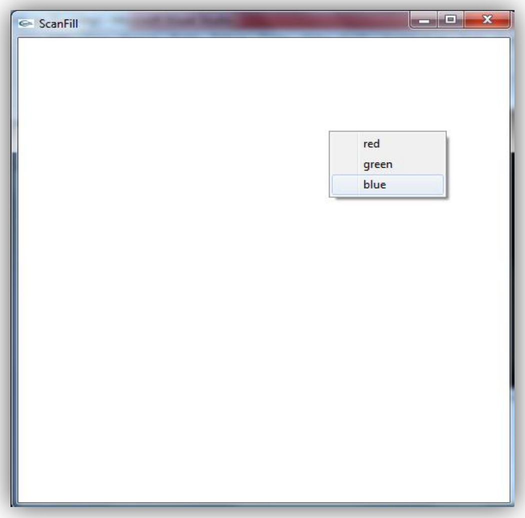
Develop a menu driven program to fill the polygon using scan line algorithm

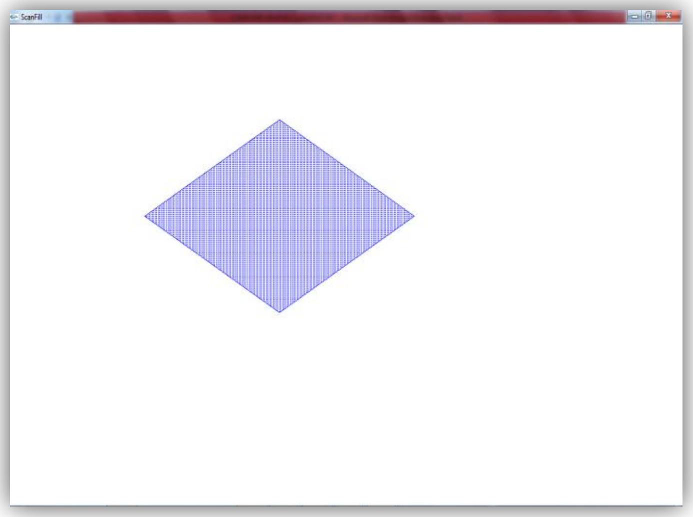
```
#include "stdafx.h"
#include<GL/glut.h>
#define BLACK 0

float x1,x2,x3,x4,y1,y2,y3,y4;

void edgedetect(float x1,float y1,float x2,float y2,int *le,int *re)
{
        float mx,x,temp;
        int i;
        if((y2-y1)<0)
        {
                temp=y1;y1=y2;y2=temp;
                temp=x1;x1=x2;x2=temp;
        }
        if((y2-y1)!=0)
                mx=(x2-x1)/(y2-y1);
        else
                mx=x2-x1;
        x=x1;
        for(i=y1;i<=y2;i++)
        {
                if(x<(float)le[i])
                        le[i]=(int)x;
                if(x>(float)re[i])
                        re[i]=(int)x;
                x+=mx;
        }
}

void draw_pixel(int x,int y,int value)
{
        glBegin(GL_POINTS);
        glVertex2i(x,y);
        glEnd();
}

void delay()
{
        int i=0;
        while(i<=1000)
                i++;
}
```

```
void scanfill(float x1,float y1,float x2,float y2,float x3,float y3,float x4,float y4)
{
    int le[500],re[500];
    int i,y;
    for(i=0;i<500;i++)
    {
        le[i]=500;
        re[i]=0;
    }
    edgedetect(x1,y1,x2,y2,le,re);
    edgedetect(x2,y2,x3,y3,le,re);
    edgedetect(x3,y3,x4,y4,le,re);
    edgedetect(x4,y4,x1,y1,le,re);
    for(y=0;y<500;y++)
    {
        if(le[y]<=re[y])
            for(i=(int)le[y];i<(int)re[y];i++)
                draw_pixel(i,y,BLACK);
          glFlush();
    }
}

void display()
{
    x1=200.0,y1=200.0,x2=100.0,y2=300.0,x3=200.0,y3=400.0,x4=300.0,y4=300.0;
    glClear(GL_COLOR_BUFFER_BIT);
    glBegin(GL_LINE_LOOP);
    glVertex2f(x1,y1);
    glVertex2f(x2,y2);
    glVertex2f(x3,y3);
    glVertex2f(x4,y4);
    glEnd();
    scanfill(x1,y1,x2,y2,x3,y3,x4,y4);
    glFlush();
}

void myInit()
{
    glClearColor(1.0,1.0,1.0,1.0);
    glPointSize(1.0);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0.0,499.0,0.0,499.0);
}
```

```
void menu(int id)
{
        switch(id)
        {
          case 0: glColor3f(1.0,0.0,0.0);
                  break;
          case 1: glColor3f(0.0,1.0,0.0);
                                  break;
          case 2: glColor3f(0.0,0.0,1.0);
                  break;
        }
        glutPostRedisplay();
}

int main(int argc,char **argv)
{
        glutInit(&argc,argv);
        glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
        glutInitWindowSize(500,500);
        glutInitWindowPosition(0,0);
        glutCreateWindow("ScanFill");
        glutDisplayFunc(display);
        glutCreateMenu(menu);
        glutAddMenuEntry("red",0);
        glutAddMenuEntry("green",1);
        glutAddMenuEntry("blue",2);
        glutAttachMenu(GLUT_LEFT_BUTTON);
        myInit();
        glutMainLoop();
        return 0;
}
```

Output:





–