```
  #include<bits/stdc++.h>
#include<GL/glut.h>
using namespace std;
bool colorcomplete=false;
bool complete=false;
int i=0;
class points{
    public:
    int x,y,z;
        float cx,cy,cz;
    void setxy(int p,int q){
        x=p;
        y=q;
    }
    void scanline(points p[],int n);
    void colorfill();
    void dda_line(int x1,int y1,int x2,int y2);
}p[20];
void init()
{
        glClearColor(0,0,0,0);
        glClear(GL_COLOR_BUFFER_BIT);
        gluOrtho2D(0,640,480,0);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
}
void points:: dda_line(int xini,int yini,int xend,int yend)
{
        glPointSize(1.0);
        double dx=(xend-xini);
        double dy=(yend-yini);
        double steps;
        float xInc,yInc,x=xini,y=yini;
        steps=(abs(dx)>abs(dy))?(abs(dx)):(abs(dy));
        xInc=dx/(float)steps;
        yInc=dy/(float)steps;
        glLineWidth(10.0);
        glVertex2f(x,y);
        int k;
        for(k=0;k<steps;k++)
        {
          x+=xInc;
          y+=yInc;
         glVertex2f(x,y);
        }
}

void points:: scanline(points p[],int n)
{
```

```
    float m[10];
    int interx[10];
    int ymax=0,ymin=500,dx,dy;

    for(int i=0;i<n;i++)
    {
        if(p[i].y>ymax)
            ymax=p[i].y;
        if(p[i].y<ymin)
            ymin=p[i].y;

        dx=p[i+1].x-p[i].x;
        dy=p[i+1].y-p[i].y;

        if(dx==0)
            m[i]=0;
        if(dy==0)
            m[i]=1;

        if(dx!=0 && dy!=0)
           {
             m[i]=(float)dx/dy;
           }
    }
    int k;
    for(int y=ymax;y>=ymin;y--)
    {
                k=0;
                for(int i=0;i<n;i++)
                {
                    if(p[i].y>y && p[i+1].y<=y || p[i].y<=y && p[i+1].y>y){
                        interx[k++]=p[i].x+(m[i]*(y-p[i].y));
                    }
                }

                sort(interx,interx+k);

                for(int i=0;i<k-1;i+=2)
                {
                    glBegin(GL_POINTS);
                    p[0].dda_line(interx[i],y,interx[i+1],y);

                    glEnd();
                    glFlush();
                }
            }
}

void points::colorfill()
{
```

```
        glClear(GL_COLOR_BUFFER_BIT);
        glPointSize(1);

        glColor3f(1,0,0);
        glRecti(500,50,550,100);

        glColor3f(0,1,0);
        glRecti(550,50,600,100);

        glColor3f(0,0,1);
        glRecti(500,100,550,150);

        glColor3f(1,0.5,0.5);
        glRecti(550,100,600,150);

        glColor3f(0,1,1);
        glRecti(500,150,550,200);

        glColor3f(1.5,0,1.5);
        glRecti(550,150,600,200);

        glColor3f(10,8,5);
        glRecti(500,200,550,250);

        glColor3f(0.5,0.5,0.5);
        glRecti(550,200,600,250);

        glFlush();
}
void mouse(int button,int state,int x,int y)
{
  if(!colorcomplete)
                p[0].colorfill();
  if(x>=500&&x<=600)
   {
        if(button==GLUT_LEFT_BUTTON && state==GLUT_DOWN)
        {
                if(x>=500&&x<=550)
                {
                        if(y>=50&&y<=100)
                        {
                                glColor3f(1,0,0);
                                colorcomplete=true;
                        }
                        else if(y>=100&&y<=150)
                        {
                                glColor3f(0,0,1);
                                colorcomplete=true;
                        }
                        else if(y>=150&&y<=200)
```

```
                        {
                                glColor3f(1,0,1);
                                colorcomplete=true;
                        }
                        else if(y>=200&&y<=250)
                        {
                                glColor3f(10,8,5);
                                colorcomplete=true;
                        }
                }else if(x>=550&&x<=600)
                {
                        if(y>=50&&y<=100)
                        {
                                colorcomplete=true;
                                glColor3f(0,1,0);
                        }
                        else if(y>=100&&y<=150)
                        {
                                colorcomplete=true;
                                glColor3f(1,0.5,0.5);
                        }
                        else if(y>=150&&y<=200)
                        {
                                colorcomplete=true;
                                glColor3f(1.5,0,1.5);
                        }
                        else if(y>=200&&y<=250)
                        {
                                colorcomplete=true;
                                glColor3f(0.5,0.5,0.5);
                        }

                }
        }
  }

  if(x>=0&&x<500)
        {
        if(colorcomplete &&!complete && button==GLUT_LEFT_BUTTON &&
state==GLUT_DOWN)
        {
                p[i++].setxy(x,y);
                glPointSize(4);
                glBegin(GL_POINTS);
            glVertex2i(x,y);
        glEnd();
        glFlush();
        }
        if(!complete && button==GLUT_RIGHT_BUTTON && state==GLUT_DOWN){
        complete=true;
```

```
        p[i].setxy(p[0].x,p[0].y);
                        glutPostRedisplay();
         }
   if(!complete && i>1)
                glutPostRedisplay();
    if(complete && button==GLUT_LEFT_BUTTON && state==GLUT_DOWN)
        p[0].scanline(p,i);
 }

}
void display()
{
        if(!complete){
        glPointSize(1);
        glBegin(GL_POINTS);
                p[0].dda_line(p[i-2].x,p[i-2].y,p[i-1].x,p[i-1].y);
        glEnd();
        glFlush();
    }
    else{
        glPointSize(1);
        glBegin(GL_POINTS);
                p[0].dda_line(p[i-1].x,p[i-1].y,p[0].x,p[0].y);
        glEnd();
        glFlush();
    }
}
int main(int argc,char *argv[])
{
        glutInit(&argc,argv);
        glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
        glutInitWindowSize(640,480);
        glutInitWindowPosition(100,100);
        glutCreateWindow("ScanLine Polygon Fill - Concave - Priya 7241");
        init();
        glutDisplayFunc(display);
        glutMouseFunc(mouse);
        glutMainLoop();
        return 0;
}
```