**/\*EXPERIMENT NO.- 5**

**PROBLEM STATEMENT-** Implement Cohen Sutherland Hodgman algorithm to clip any given polygon. Provide the vertices of the polygon to be clipped and pattern of clipping interactively.

**ROLL NO.-** ITSA01

**NAME-** Rahul Devidas Thakare

\*/ **INPUT**

#include<iostream>

#include<GL/gl.h>

#include<GL/glu.h>

#include<GL/glut.h>

#include<math.h>

using namespace std;

typedef struct{

    float x;

    float y;

}PT;

int n;

PT d,p1,p2,p[20],pi1,pi2,pp[20];

void left(){

    int i=0;

    int j=0;

    for(i=0;i<n;i++){

        if(p[i].x<p1.x&&p[i+1].x>=p1.x){

            if(p[i+1].x-p[i].x!=0){

                pp[j].y=(p[i+1].y-p[i].y)/(p[i+1].x-p[i].x)\*(p1.x-p[i].x)+p[i].y;

            }

            else{

                pp[j].y=p[i].y;

            }

            pp[j].x=p1.x;

            j++;

            pp[j].x=p[i+1].x;

            pp[j].y=p[i+1].y;

            j++;

        }

        if(p[i].x>=p1.x&&p[i+1].x>=p1.x){

            pp[j].y=p[i+1].y;

            pp[j].x=p[i+1].x;

            j++;

        }

        if(p[i].x>=p1.x&&p[i+1].x<p1.x){

            if(p[i+1].x-p[i].x!=0){

                pp[j].y=(p[i+1].y-p[i].y)/(p[i+1].x-p[i].x)\*(p1.x-p[i].x)+p[i].y;

            }

            else{

                pp[j].y=p[i].y;

            }

            pp[j].x=p1.x;

            j++;

        }

    }

    for(i=0;i<j;i++){

        p[i].x=pp[i].x;

        p[i].y=pp[i].y;

    }

    p[i].x=pp[0].x;

    p[i].y=pp[0].y;

    n=j;

}

void right(){

    int i=0;

    int j=0;

    for(i=0;i<n;i++){

        if(p[i].x>p2.x&&p[i+1].x<=p2.x){

            if(p[i+1].x-p[i].x!=0){

                pp[j].y=(p[i+1].y-p[i].y)/(p[i+1].x-p[i].x)\*(p2.x-p[i].x)+p[i].y;

            }

            else{

                pp[j].y=p[i].y;

            }

            pp[j].x=p2.x;

            j++;

            pp[j].x=p[i+1].x;

            pp[j].y=p[i+1].y;

            j++;

        }

        if(p[i].x<=p2.x&&p[i+1].x<=p2.x){

            pp[j].y=p[i+1].y;

            pp[j].x=p[i+1].x;

            j++;

        }

        if(p[i].x<=p2.x&&p[i+1].x>p2.x){

            if(p[i+1].x-p[i].x!=0){

                pp[j].y=(p[i+1].y-p[i].y)/(p[i+1].x-p[i].x)\*(p2.x-p[i].x)+p[i].y;

            }

            else{

                pp[j].y=p[i].y;

            }

            pp[j].x=p2.x;

            j++;

        }

    }

    for(i=0;i<j;i++){

        p[i].x=pp[i].x;

        p[i].y=pp[i].y;

    }

    p[i].x=pp[0].x;

    p[i].y=pp[0].y;

    n=j;

}

void top(){

    int i=0;

    int j=0;

    for(i=0;i<n;i++){

        if(p[i].y>p2.y&&p[i+1].y<=p2.y){

            if(p[i+1].y-p[i].y!=0){

                pp[j].x=(p[i+1].x-p[i].x)/(p[i+1].y-p[i].y)\*(p2.y-p[i].y)+p[i].x;

            }

            else{

                pp[j].x=p[i].x;

            }

            pp[j].y=p2.y;

            j++;

            pp[j].x=p[i+1].x;

            pp[j].y=p[i+1].y;

            j++;

        }

        if(p[i].y<=p2.y&&p[i+1].y<=p2.y){

            pp[j].y=p[i+1].y;

            pp[j].x=p[i+1].x;

            j++;

        }

        if(p[i].y<=p2.y&&p[i+1].y>p2.y){

            if(p[i+1].y-p[i].y!=0){

                pp[j].x=(p[i+1].x-p[i].x)/(p[i+1].y-p[i].y)\*(p2.y-p[i].y)+p[i].x;

            }

            else{

                pp[j].x=p[i].x;

            }

            pp[j].x=p2.y;

            j++;

        }

    }

    for(i=0;i<j;i++){

        p[i].x=pp[i].x;

        p[i].y=pp[i].y;

    }

    p[i].x=pp[0].x;

    p[i].y=pp[0].y;

    n=j;

}

void bottom(){

    int i=0;

    int j=0;

    for(i=0;i<n;i++){

        if(p[i].y<p1.y&&p[i+1].y>=p1.y){

            if(p[i+1].y-p[i].y!=0){

                pp[j].x=(p[i+1].x-p[i].x)/(p[i+1].y-p[i].y)\*(p1.y-p[i].y)+p[i].x;

            }

            else{

                pp[j].x=p[i].x;

            }

            pp[j].y=p1.y;

            j++;

            pp[j].x=p[i+1].x;

            pp[j].y=p[i+1].y;

            j++;

        }

        if(p[i].y>=p1.y&&p[i+1].y>=p1.y){

            pp[j].y=p[i+1].y;

            pp[j].x=p[i+1].x;

            j++;

        }

        if(p[i].y>=p1.y&&p[i+1].y,p1.y){

            if(p[i+1].y-p[i].y!=0){

                pp[j].x=(p[i+1].x-p[i].x)/(p[i+1].y-p[i].y)\*(p1.y-p[i].y)+p[i].x;

            }

            else{

                pp[j].x=p[i].x;

            }

            pp[j].x=p1.y;

            j++;

        }

    }

    for(i=0;i<j;i++){

        p[i].x=pp[i].x;

        p[i].y=pp[i].y;

    }

    p[i].x=pp[0].x;

    p[i].y=pp[0].y;

    n=j;

}

void drawpolygon(){

    int i;

    glColor3f(1.0,0.0,0.0);

    for(i=0;i<n-1;i++){

        glBegin(GL\_LINES);

            glVertex2d(p[i].x,p[i].y);

            glVertex2d(p[i+1].x,p[i+1].y);

        glEnd();

    }

    glBegin(GL\_LINES);

            glVertex2d(p[i].x,p[i].y);

            glVertex2d(p[0].x,p[0].y);

    glEnd();

}

void myMouse(int button,int state,int x,int y){

    if(button==GLUT\_LEFT\_BUTTON&&state==GLUT\_DOWN){

        glClear(GL\_COLOR\_BUFFER\_BIT);

        glBegin(GL\_LINE\_LOOP);

            glVertex2f(p1.x,p1.y);

            glVertex2f(p2.x,p1.y);

            glVertex2f(p2.x,p2.y);

            glVertex2f(p1.x,p2.y);

        glEnd();

        left();

        right();

        top();

        bottom();

        drawpolygon();

    }

    glFlush();

}

void display(void){

    glClear(GL\_COLOR\_BUFFER\_BIT);

    glColor3f(0.4,1.0,0.0);

    glBegin(GL\_LINE\_LOOP);

        glVertex2f(p1.x,p1.y);

        glVertex2f(p2.x,p1.y);

        glVertex2f(p2.x,p2.y);

        glVertex2f(p1.x,p2.y);

    glEnd();

    drawpolygon();

    glFlush();

}

void init(void){

    glClearColor(0.0,0.0,0.0,0.0);

    gluOrtho2D(0,500,0,500);

}

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

    int i;

    cout<<"\nEnter co-ordinates(left,bottom)-";

    cout<<"\nEnter x-coordinates-";

    cin>>p1.x;

    cout<<"\nEnter y-coordinates-";

    cin>>p1.y;

    cout<<"\nEnter co-ordinates(right,top)-";

    cout<<"\nEnter x-coordinates-";

    cin>>p2.x;

    cout<<"\nEnter y-coordinates-";

    cin>>p2.y;

    cout<<"\nEnter the number of Vertices-";

    cin>>n;

    for(i=0;i<n;i++){

        cout<<"\nEnter co-ordinates of Vertex-";

        cout<<i+1;

        cout<<"\nEnter x-coordinates-";

        cin>>p[i].x;

        cout<<"\nEnter y-coordinates-";

        cin>>p[i].y;

    }

    p[i].x=p[0].x;

    p[i].y=p[0].y;

    glutInit(&argc,argv);

    glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB);

    glutInitWindowSize(500,500);

    glutInitWindowPosition(0,0);

    glutCreateWindow("Sutherland Hodgeman Polygon Clipping");

    init();

    glutDisplayFunc(display);

    glutMouseFunc(myMouse);

    glFlush();

    glutMainLoop();

    return 0;

}