8) Write a program to implement the Cohen-Hodgeman polygon clipping algorithm. Make provision to specify the input polygon and window for clipping.

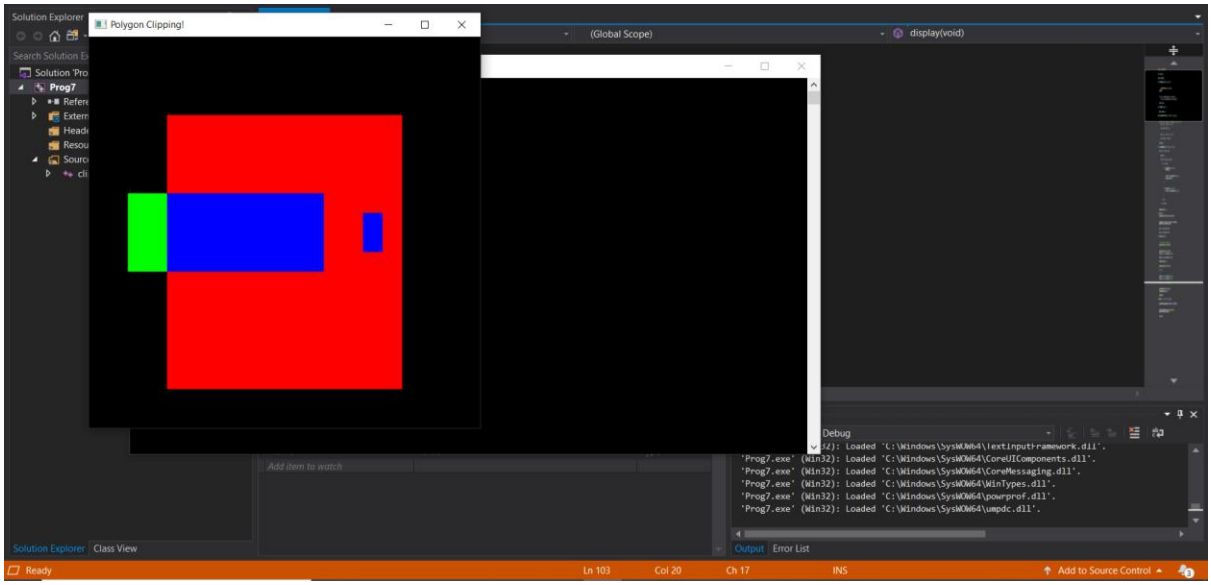
```
#include
<windows.h>
#include <gl/glut.h>
struct Point {
	float x, y;
} w[4], oVer[4];
int Nout;
void drawPoly(Point p[], int n) {
	glBegin(GL_POLYGON);
	for (int i = 0;i < n;i++)
		glVertex2f(p[i].x, p[i].y);
	glEnd();
}
bool insideVer(Point p) {
	if ((p.x >= w[0].x) && (p.x <= w[2].x))
		if ((p.y >= w[0].y) && (p.y <= w[2].y))
			return true;
	return false;
}
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


        glColor3f(0.0f, 1.0f, 0.0f);
        inVer[0].x = 70, inVer[0].y = 45;
        inVer[1].x = 70, inVer[1].y = 55;
        inVer[2].x = 75, inVer[2].y = 55;
        inVer[3].x = 75, inVer[3].y = 45;
        drawPoly(inVer, 4);
        // As Rect
        glColor3f(0.0f, 0.0f, 1.0f);
        clipAndDraw(inVer, 4);


        glFlush();
}
int main(int argc, char* argv[]) {
        glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
        glutInitWindowSize(500, 500);
        glutInitWindowPosition(100, 100);
        glutCreateWindow("Polygon Clipping!");
        glutDisplayFunc(display);
        glutMainLoop();
        return 0;
}
```


Polygon Clipping!
Solution Explorer
(Global Scope)
display(void)
Prog7
Debug
Add item to watch
Solution Explorer
Class View
Output
Error List
Ready
Ln 103
Col 20
Ch 17
INS
Add to Source Control