Name	**:**	**V Vikram**	**Class**	**:** CSE ‘C’
Reg. No.	**:**	**18 5001 194**	**Date**	**:** 13/09/2021
Subject	**:**	**UCS1712---Graphics and Multimedia Lab**		

QUESTION :

Lab Exercise 7

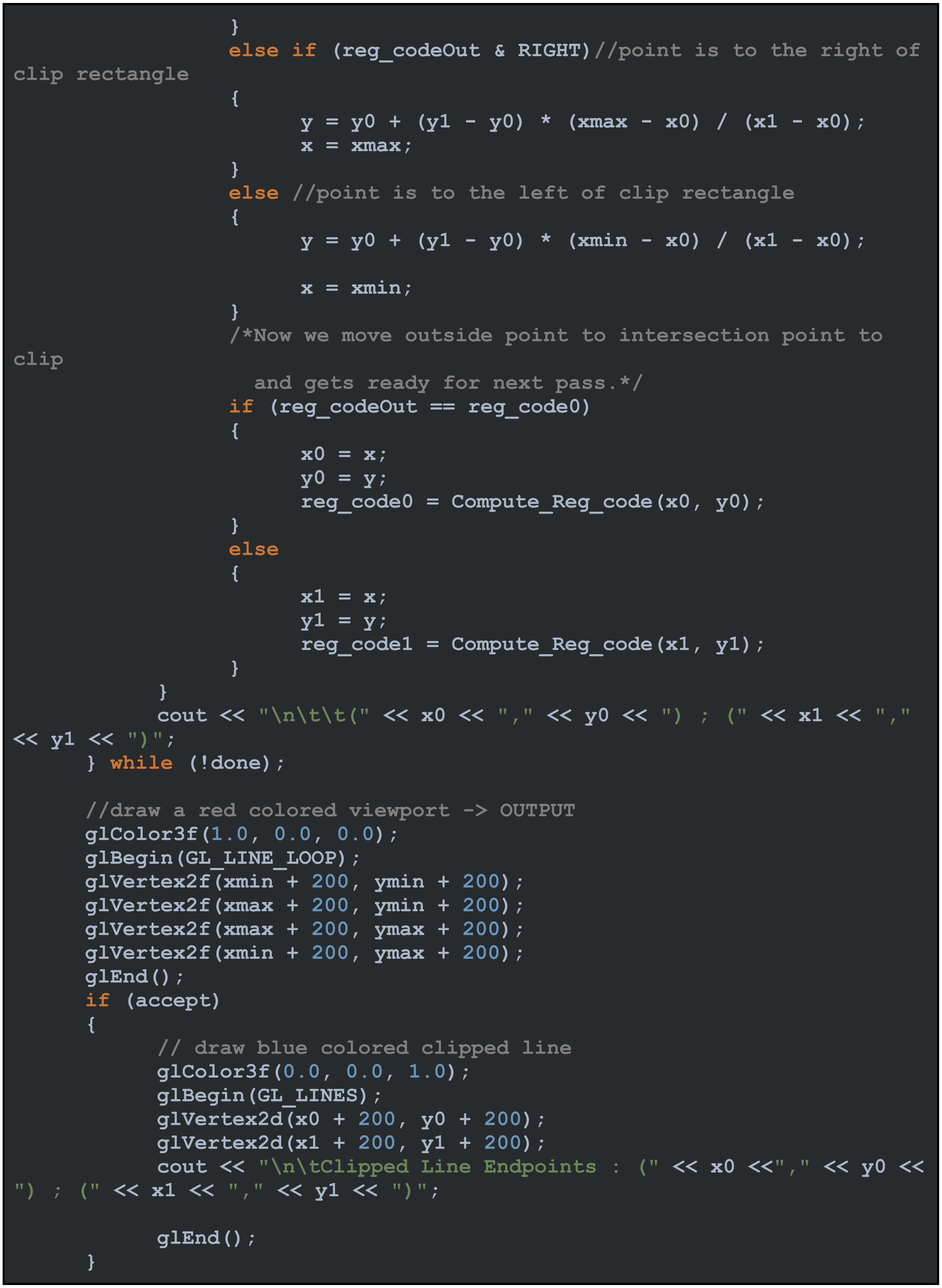
Cohen Sutherland Line clipping in C++ using OpenGL

Apply Cohen Sutherland line clipping on a
line : (x1,y1) (x2,y2) with respect to a
clipping window : (XWmin,YWmin) (XWmax,YWmax).

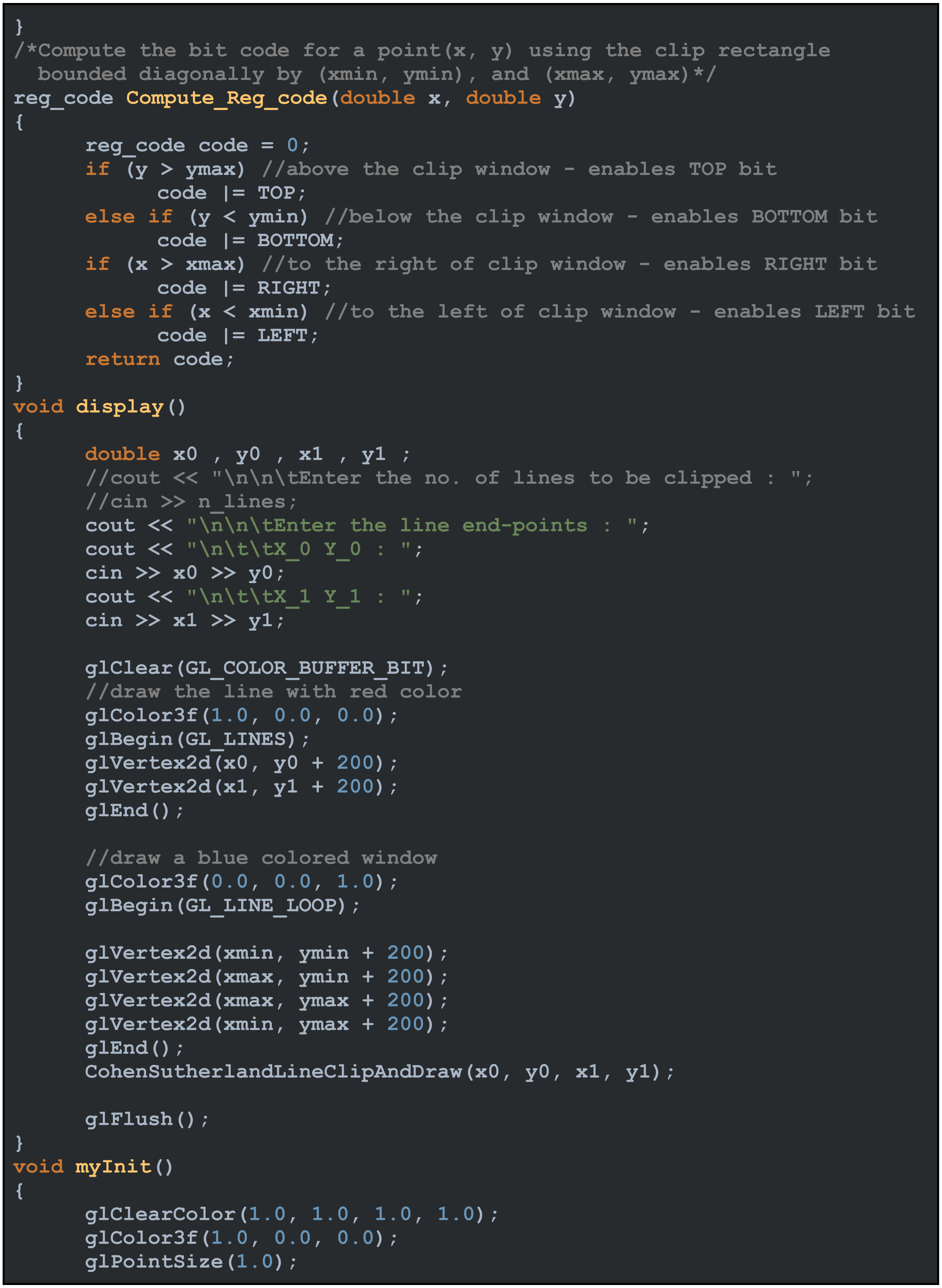
CODE :-

Main.cpp :

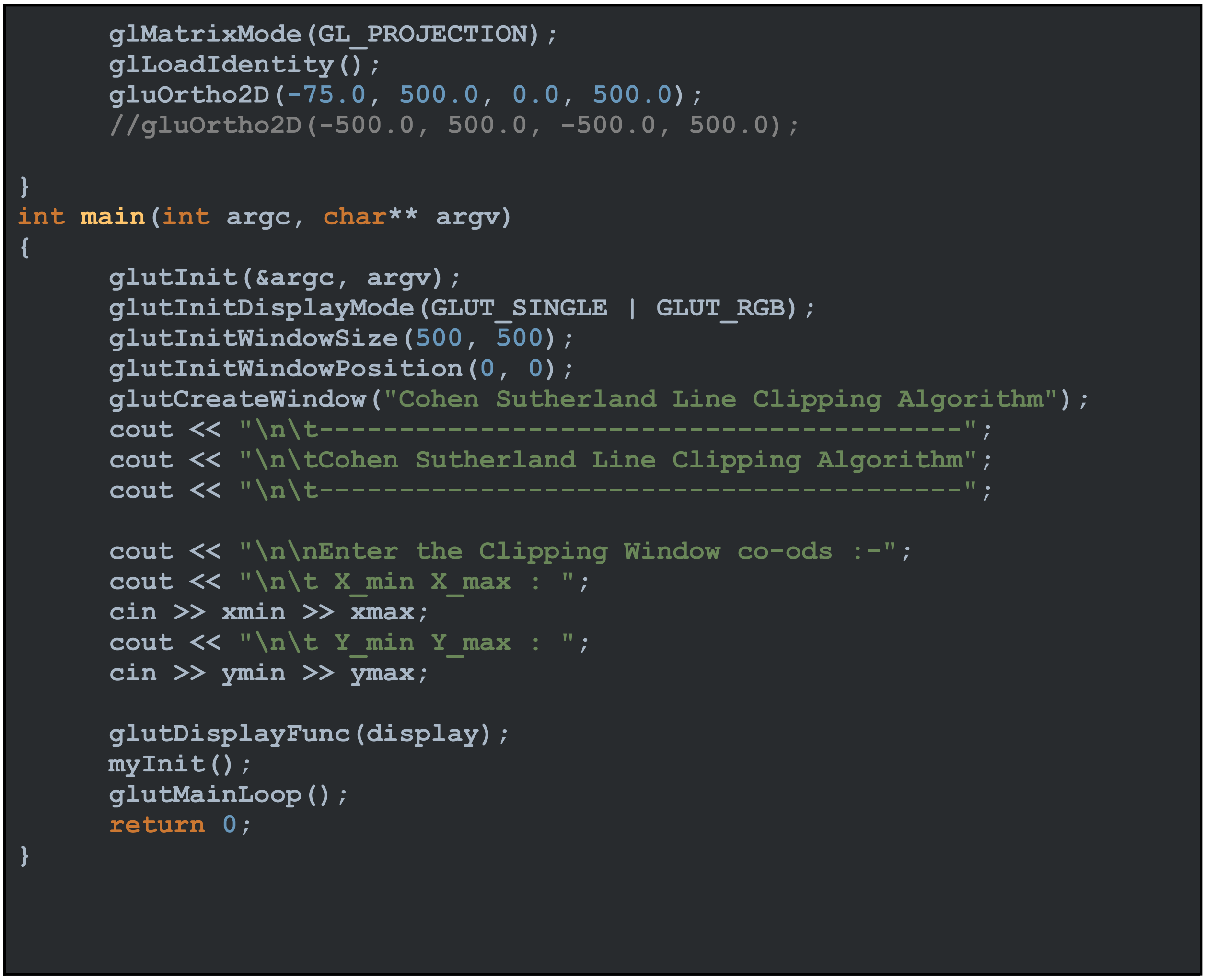
```
/* Cohen-Sutherland Line Clipping Algorithm with Window to viewport
Mapping */
#include <stdio.h>
#include <GL/glut.h>
#include <iostream>
using namespace std;

#define reg_code int
double xmin, ymin, xmax, ymax; // Window boundaries
//int n_lines;
//bit codes for the top,bottom,right & left
const int TOP = 8;
const int BOTTOM = 4;
const int RIGHT = 2;
const int LEFT = 1;
//used to compute bit codes of a point
reg_code Compute_Reg_code(double x, double y);
/*Cohen - Sutherland clipping algorithm clips a line from P0 = (x0, y0)
to P1 = (x1, y1)
  against a rectangle with diagonal from (xmin, ymin) to (xmax, ymax).*/


void CohenSutherlandLineClipAndDraw(double x0, double y0, double x1,
double y1)
{
      //reg_codes for P0, P1, and whatever point lies outside the clip
rectangle
      reg_code reg_code0, reg_code1, reg_codeOut;
      bool accept = false, done = false;
      //compute RegionCodes(RC) for the endpoints
      reg_code0 = Compute_Reg_code(x0, y0);
      reg_code1 = Compute_Reg_code(x1, y1);
      cout << "\n\tIntermediate Endpoints:-";
      do
      {
            if (!(reg_code0 | reg_code1)) //if(RC==0000) -> Trivially
accept & exit
            {
                  accept = true;

                  done = true;
            }
            else if (reg_code0 & reg_code1) // if(RC==1111) -> Trivially
reject & exit
                  done = true;
            else
            { /*failed both tests, so calculate the line segment to clip
from an
                  outside point to an intersection with clip edge*/
                  double x, y;
                  //At least one endpoint is outside the clip rectangle;
pick it.
                  reg_codeOut = reg_code0 ? reg_code0 : reg_code1;
                  /*Now find the intersection point;
                    use formulas y = y0 + slope * (x - x0) ; x = x0 +
(1/slope)* (y - y0)*/
                  if (reg_codeOut & TOP) //point is above the clip
rectangle
                  {
                        x = x0 + (x1 - x0) * (ymax - y0) / (y1 - y0);
                        y = ymax;
                  }
                  else if (reg_codeOut & BOTTOM) //point is below the
clip rectangle
                  {
                        x = x0 + (x1 - x0) * (ymin - y0) / (y1 - y0);
                        y = ymin;
```



```
            }
            else if (reg_codeOut & RIGHT)//point is to the right of clip rectangle
            {
                y = y0 + (y1 - y0) * (xmax - x0) / (x1 - x0);
                x = xmax;
            }
            else //point is to the left of clip rectangle
            {
                y = y0 + (y1 - y0) * (xmin - x0) / (x1 - x0);

                x = xmin;
            }
            /*Now we move outside point to intersection point to clip
              and gets ready for next pass.*/
            if (reg_codeOut == reg_code0)
            {
                x0 = x;
                y0 = y;
                reg_code0 = Compute_Reg_code(x0, y0);
            }
            else
            {
                x1 = x;
                y1 = y;
                reg_code1 = Compute_Reg_code(x1, y1);
            }
        }
        cout << "\n\t\t(" << x0 << "," << y0 << ") ; (" << x1 << "," << y1 << ")";
    } while (!done);

    //draw a red colored viewport -> OUTPUT
    glColor3f(1.0, 0.0, 0.0);
    glBegin(GL_LINE_LOOP);
    glVertex2f(xmin + 200, ymin + 200);
    glVertex2f(xmax + 200, ymin + 200);
    glVertex2f(xmax + 200, ymax + 200);
    glVertex2f(xmin + 200, ymax + 200);
    glEnd();
    if (accept)
    {
        // draw blue colored clipped line
        glColor3f(0.0, 0.0, 1.0);
        glBegin(GL_LINES);
        glVertex2d(x0 + 200, y0 + 200);
        glVertex2d(x1 + 200, y1 + 200);
        cout << "\n\tClipped Line Endpoints : (" << x0 <<"," << y0 << ") ; (" << x1 << "," << y1 << ")";

        glEnd();
    }
```



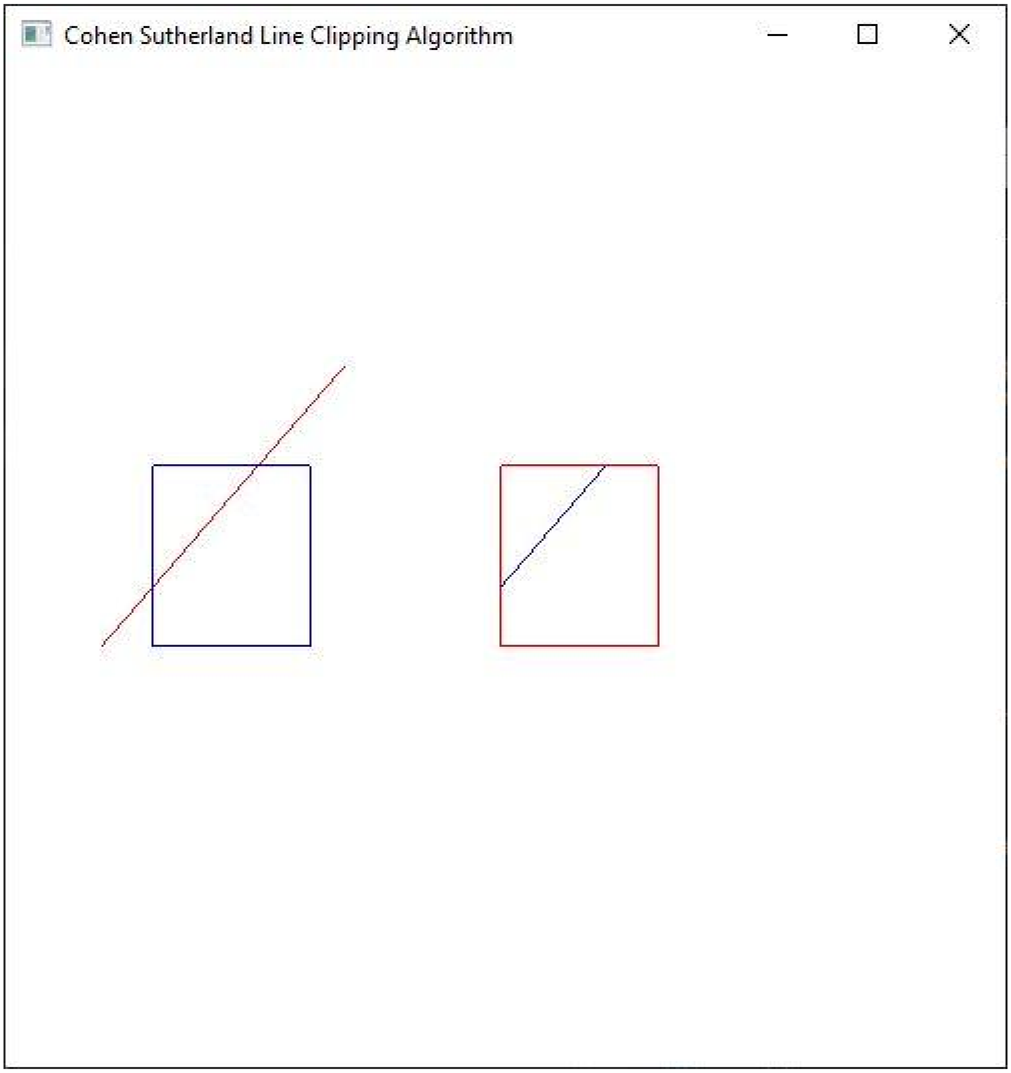
```
}
/*Compute the bit code for a point(x, y) using the clip rectangle
  bounded diagonally by (xmin, ymin), and (xmax, ymax)*/
reg_code Compute_Reg_code(double x, double y)
{
      reg_code code = 0;
      if (y > ymax) //above the clip window - enables TOP bit
            code |= TOP;
      else if (y < ymin) //below the clip window - enables BOTTOM bit
            code |= BOTTOM;
      if (x > xmax) //to the right of clip window - enables RIGHT bit
            code |= RIGHT;
      else if (x < xmin) //to the left of clip window - enables LEFT bit
            code |= LEFT;
      return code;
}
void display()
{
      double x0 , y0 , x1 , y1 ;
      //cout << "\n\n\tEnter the no. of lines to be clipped : ";
      //cin >> n_lines;
      cout << "\n\n\tEnter the line end-points : ";
      cout << "\n\t\tX_0 Y_0 : ";
      cin >> x0 >> y0;
      cout << "\n\t\tX_1 Y_1 : ";
      cin >> x1 >> y1;

      glClear(GL_COLOR_BUFFER_BIT);
      //draw the line with red color
      glColor3f(1.0, 0.0, 0.0);
      glBegin(GL_LINES);
      glVertex2d(x0, y0 + 200);
      glVertex2d(x1, y1 + 200);
      glEnd();

      //draw a blue colored window
      glColor3f(0.0, 0.0, 1.0);
      glBegin(GL_LINE_LOOP);

      glVertex2d(xmin, ymin + 200);
      glVertex2d(xmax, ymin + 200);
      glVertex2d(xmax, ymax + 200);
      glVertex2d(xmin, ymax + 200);
      glEnd();
      CohenSutherlandLineClipAndDraw(x0, y0, x1, y1);

      glFlush();
}
void myInit()
{
      glClearColor(1.0, 1.0, 1.0, 1.0);
      glColor3f(1.0, 0.0, 0.0);
      glPointSize(1.0);
```

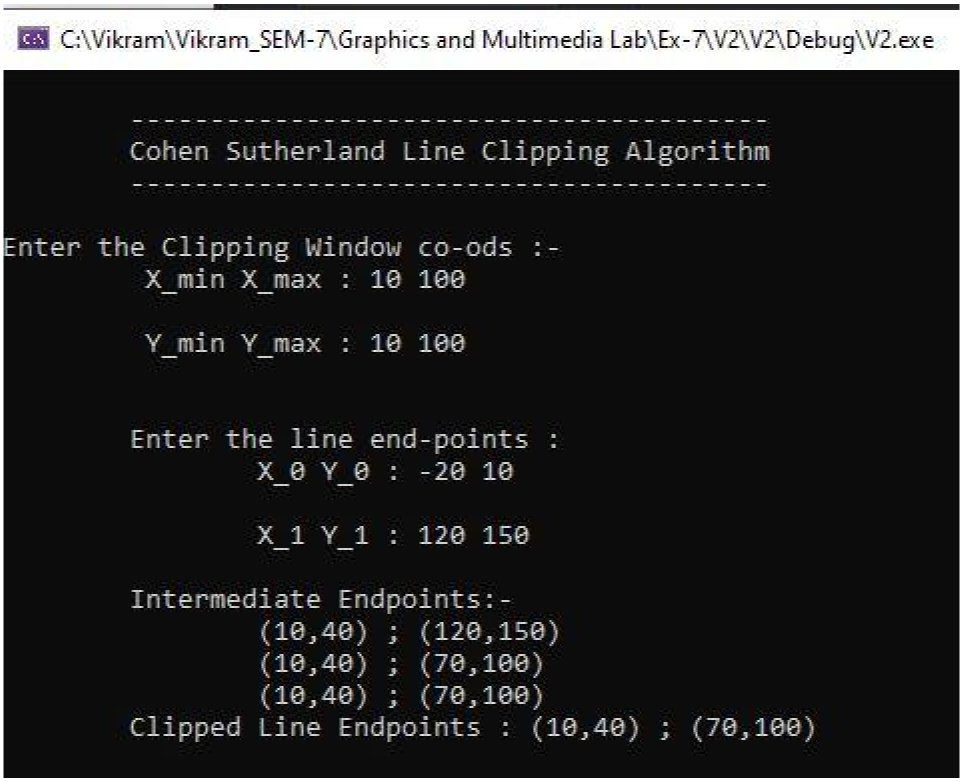


```
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	gluOrtho2D(-75.0, 500.0, 0.0, 500.0);
	//gluOrtho2D(-500.0, 500.0, -500.0, 500.0);

}
int main(int argc, char** argv)
{
	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
	glutInitWindowSize(500, 500);
	glutInitWindowPosition(0, 0);
	glutCreateWindow("Cohen Sutherland Line Clipping Algorithm");
	cout << "\n\t----------------------------------------";
	cout << "\n\tCohen Sutherland Line Clipping Algorithm";
	cout << "\n\t----------------------------------------";

	cout << "\n\nEnter the Clipping Window co-ods :-";
	cout << "\n\t X_min X_max : ";
	cin >> xmin >> xmax;
	cout << "\n\t Y_min Y_max : ";
	cin >> ymin >> ymax;

	glutDisplayFunc(display);
	myInit();
	glutMainLoop();
	return 0;
}
```

OUTPUT SNAPSHOTS :

Partially inside :

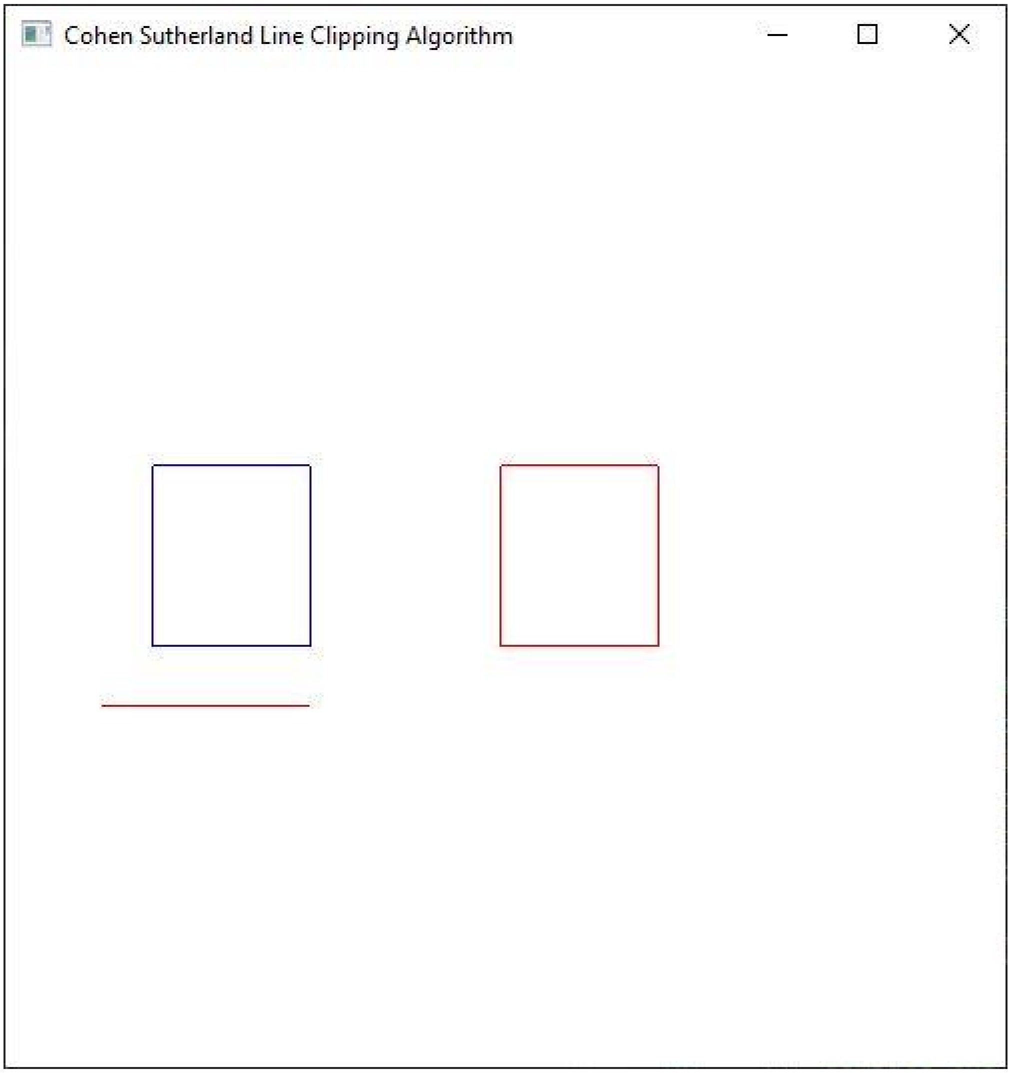


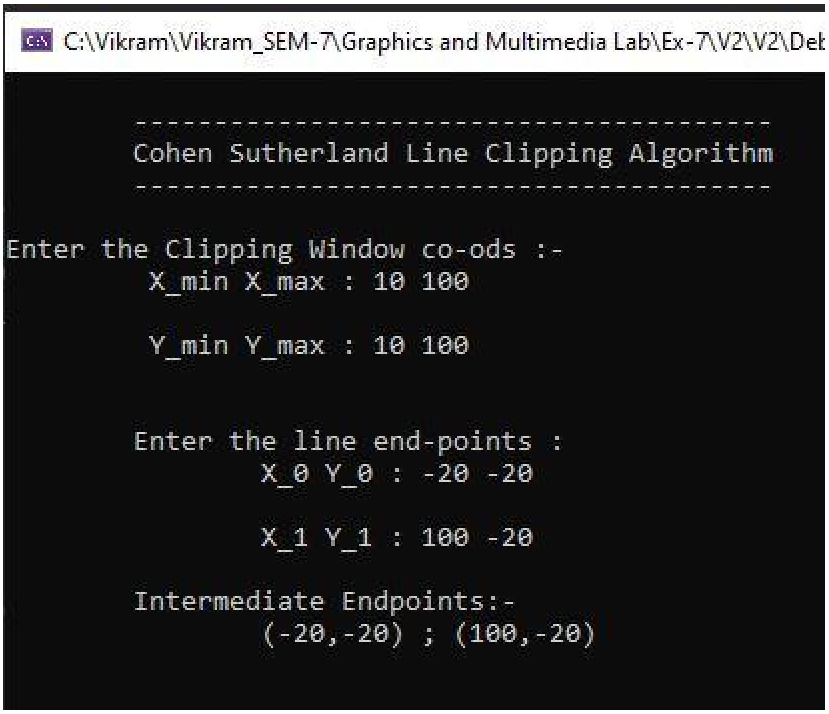


C:\Vikram\Vikram_SEM-7\Graphics and Multimedia Lab\Ex-7\V2\V2\Debug\V2.exe

```
        ----------------------------------------
        Cohen Sutherland Line Clipping Algorithm
        ----------------------------------------

Enter the Clipping Window co-ods :-
         X_min X_max : 10 100

         Y_min Y_max : 10 100


        Enter the line end-points :
                X_0 Y_0 : -20 10

                X_1 Y_1 : 120 150

        Intermediate Endpoints:-
                (10,40) ; (120,150)
                (10,40) ; (70,100)
                (10,40) ; (70,100)
        Clipped Line Endpoints : (10,40) ; (70,100)
```

Completely outside :

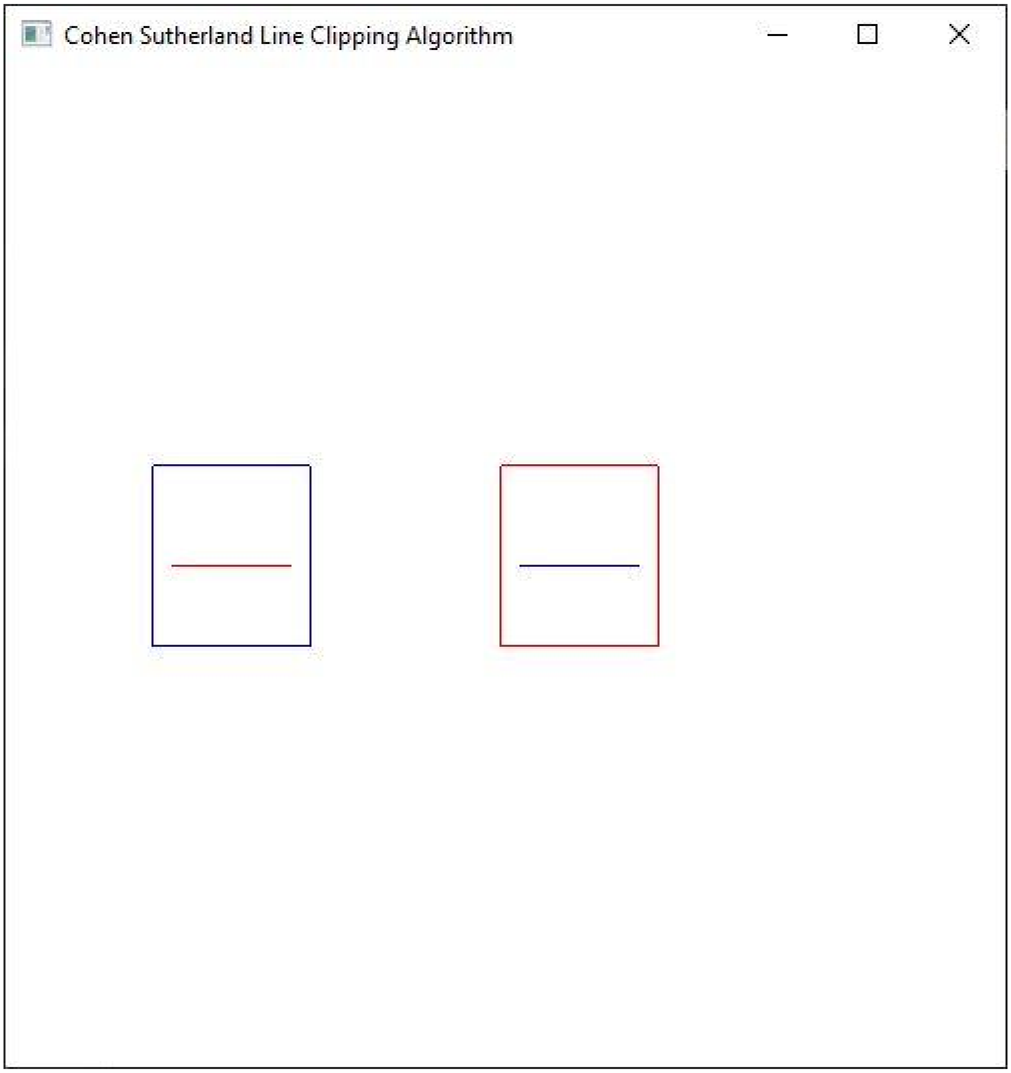


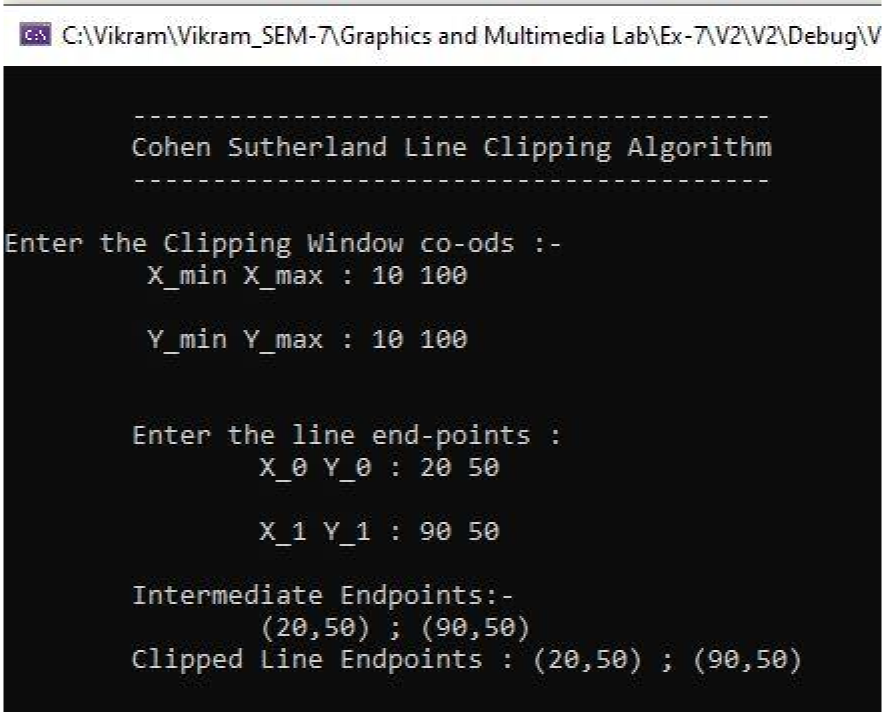


C:\Vikram\Vikram_SEM-7\Graphics and Multimedia Lab\Ex-7\V2\V2\Deb

```
        ----------------------------------------
        Cohen Sutherland Line Clipping Algorithm
        ----------------------------------------

Enter the Clipping Window co-ods :-
         X_min X_max : 10 100

         Y_min Y_max : 10 100


        Enter the line end-points :
                X_0 Y_0 : -20 -20

                X_1 Y_1 : 100 -20

        Intermediate Endpoints:-
                (-20,-20) ; (100,-20)
```

Completely inside :




C:\Vikram\Vikram_SEM-7\Graphics and Multimedia Lab\Ex-7\V2\V2\Debug\V

```
        ----------------------------------------
        Cohen Sutherland Line Clipping Algorithm
        ----------------------------------------

Enter the Clipping Window co-ods :-
         X_min X_max : 10 100

         Y_min Y_max : 10 100


        Enter the line end-points :
                X_0 Y_0 : 20 50

                X_1 Y_1 : 90 50

        Intermediate Endpoints:-
                (20,50) ; (90,50)
        Clipped Line Endpoints : (20,50) ; (90,50)
```

CONCLUSION :

Thus the Cohen Sutherland line clipping algorithm was applied on a line : (x1,y1) (x2,y2) was clipped with respect to a clipping window : (XWmin,YWmin) (XWmax,YWmax).