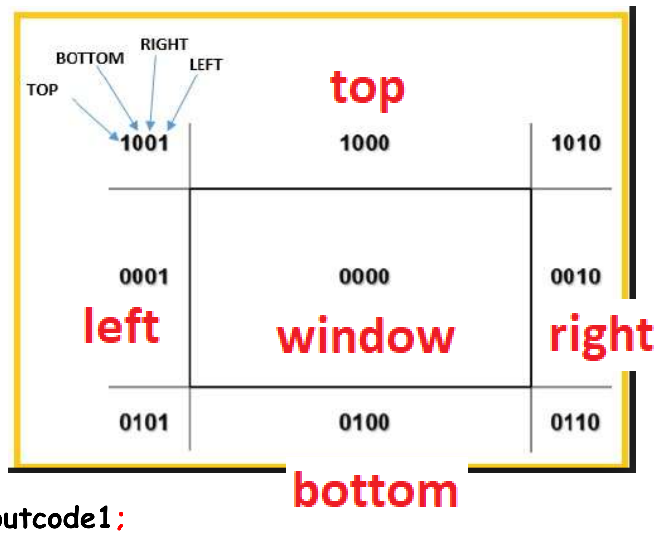
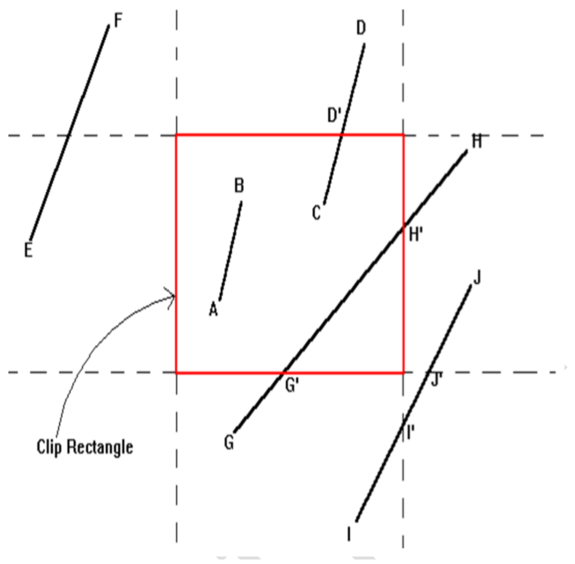
5. Program to clip a line using Cohen-Sutherland line-clipping algorithm.

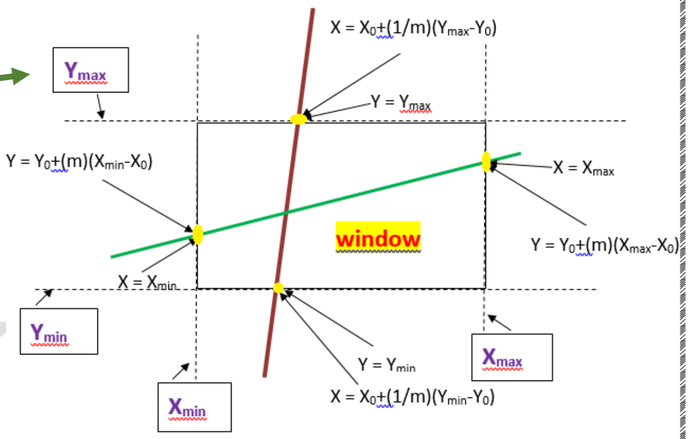
```
#include <stdio.h>
#include <GL/glut.h>
double xmin = 50, ymin = 50, xmax = 100, ymax = 100;          //window coordinates
double xvmin = 200, yvmin = 200, xvmax = 300, yvmax = 300;  //viewport coordinates

const int LEFT = 1;            // assuming code words for LEFT, RIGHT, BOTTOM & TOP.
const int RIGHT = 2;
const int BOTTOM = 4;
const int TOP = 8;

int ComputeOutCode (double x, double y)
{
    int code = 0;
    if (y > ymax)                    //above the clip window
        code |= TOP;
    else if (y < ymin)              //below the clip window
        code |= BOTTOM;
    if (x > xmax)                    //to the right of clip window
        code |= RIGHT;
    else if (x < xmin)            //to the left of clip window
        code |= LEFT;
    return code;                     //return the calculated code
}

void CohenSutherland(double x0, double y0, double x1, double y1)
{
    int outcode0, outcode1, outcodeOut;
    bool accept = false, done = false;
    outcode0 = ComputeOutCode (x0, y0);     //calculate the region of 1st point
    outcode1 = ComputeOutCode (x1, y1);      //calculate the region of 2nd point

    do
    {
        if ( ! (outcode0 | outcode1))
        {
            accept = true; //both the points
            done = true;   are inside the window
        }
        else if (outcode0 & outcode1)
            done = true; //both are outside
        else
        {
            double x, y;
            double m = (y1 - y0) / (x1 - x0);
            outcodeOut = outcode0 ? outcode0: outcode1;
```



```
            if (outcodeOut & TOP)
            {
                x = x0 + (1/m) * (ymax - y0);
                y = ymax;
            }
            else if (outcodeOut & BOTTOM)
            {
                x = x0 + (1/m) * (ymin - y0);
                y = ymin;
            }
            else if (outcodeOut & RIGHT)
            {
                y = y0 +  m * (xmax - x0);
                x = xmax;
            }
            else
            {
                y = y0 + m * (xmin - x0);
                x = xmin;
            }
            /* Intersection calculations are done,
go ahead and mark the clipped line */

            if (outcodeOut == outcode0)
            {
                x0 = x;
                y0 = y;
             outcode0 = ComputeOutCode (x0, y0);
            }
            else
            {
                x1 = x;
                y1 = y;
                outcode1 = ComputeOutCode (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;
```



Calculating Intersection Points

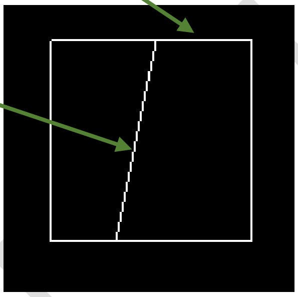


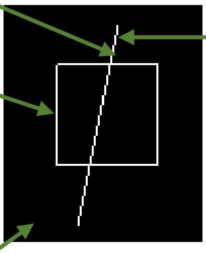
Zooming (scaling) the clipping rectangle and the clipped line and show it to the customer. The customer can see both before and after clipping effects. See the output for better clarity.

sx, sy -> scaling parameters

vx0, vy0, vx1, vy1 -> line coordinates

```
        glBegin(GL_LINE_LOOP);                // draw the zoomed rectangle
            glVertex2f (xvmin, yvmin);
            glVertex2f (xvmax, yvmin);
            glVertex2f (xvmax, yvmax);
            glVertex2f (xvmin, yvmax);
        glEnd();

        glBegin(GL_LINES);                    // draw the zoomed clipped line
            glVertex2d (vx0, vy0);
            glVertex2d (vx1, vy1);
        glEnd();
    }
}

void display()
{
    double x0 = 60, y0 = 20, x1 = 80, y1 = 120; // the line coordinates
    glClear (GL_COLOR_BUFFER_BIT);

    glColor3f(1, 1, 1);              // white colour to draw line

    glBegin (GL_LINES);
        glVertex2d (x0, y0);         // draw the line that has to be clipped
        glVertex2d (x1, y1);
    glEnd ();

   glBegin (GL_LINE_LOOP);           // draw the clipping / viewing rectangle window
        glVertex2f (xmin, ymin);
        glVertex2f (xmax, ymin);
        glVertex2f (xmax, ymax);
        glVertex2f (xmin, ymax);
    glEnd ();

    CohenSutherland (x0, y0, x1, y1);      // call the algorithm

    glFlush ();                            // show the output
}

void init()
{
    glClearColor (0, 0, 0, 1);             //black background colour
    gluOrtho2D (0, 500, 0, 500);
}
```

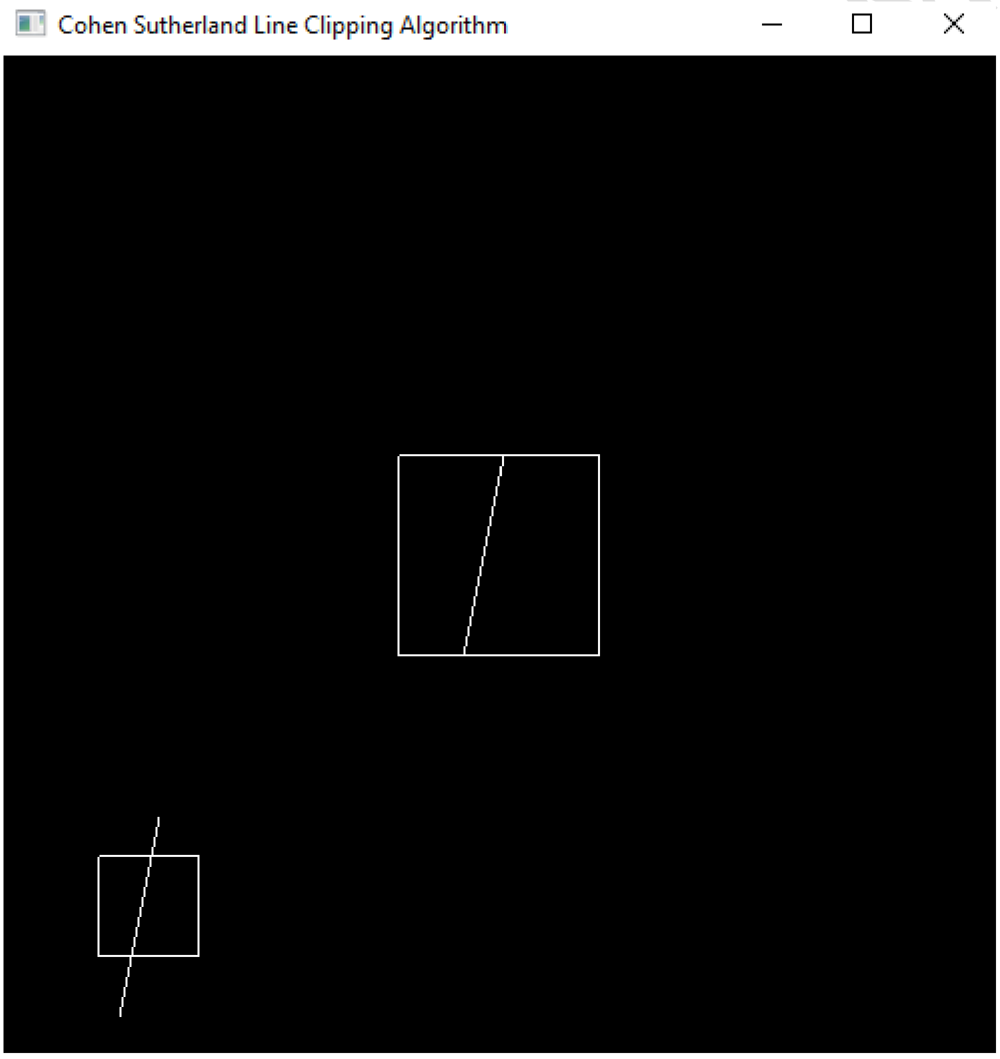




```
int main(int argc, char **argv)
{
    glutInit (&argc,argv);
    glutInitDisplayMode (GLUT_SINGLE|GLUT_RGB);
    glutInitWindowSize (500, 500);
    glutInitWindowPosition (0, 0);
    glutCreateWindow ("Cohen Sutherland Line Clipping Algorithm");

    init();

    glutDisplayFunc(display);

    glutMainLoop();
}
```

****

OUTPUT


Cohen Sutherland Line Clipping Algorithm