```
#include<stdio.h>
#include<stdlib.h>
#include<gl/glut.h>

#define outcode int
#define true 1
#define false 0
double xmin, ymin, xmax, ymax;
double xvmin, yvmin, xvmax, yvmax;


const int RIGHT = 4;
const int LEFT = 8;
const int TOP = 1;
const int BOTTOM = 2;

int n;
struct line_segment {
        int x1;
        int y1;
        int x2;
        int y2;
};
struct line_segment ls[10];

outcode computeoutcode(double x, double y)
{
        outcode code = 0;
        if (y > ymax)
                code |= TOP;
        else if (y < ymin)
                code |= BOTTOM;
        if (x > xmax)
                code |= RIGHT;
        else if (x < xmin)
                code |= LEFT;

        return code;
}

void cohensuther(double x0, double y0, double x1, double y1)
{
        outcode outcode0, outcode1, outcodeout;
        bool accept = false, done = false;

        outcode0 = computeoutcode(x0, y0);
        outcode1 = computeoutcode(x1, y1);

        do
        {
                if (!(outcode0 | outcode1))
                {
                        accept = true;
                        done = true;
                }
                else if (outcode0 & outcode1)
```

```
                        done = true;
                }
                else
                {
                        double x, y;
                        outcodeout = outcode0 ? outcode0 : outcode1;
                        if (outcodeout & TOP)
                        {
                                x = x0 + (x1 - x0) * (ymax - y0) / (y1
- y0);
                                y = ymax;
                        }
                        else if (outcodeout & BOTTOM)
                        {
                                x = x0 + (x1 - x0) * (ymin - y0) / (y1
- y0);
                                y = ymin;
                        }
                        else if (outcodeout & RIGHT)
                        {
                                y = y0 + (y1 - y0) * (xmax - x0) / (x1
- x0);
                                x = xmax;
                        }
                        else
                        {
                                y = y0 + (y1 - y0) * (xmin - x0) / (x1
- x0);
                                x = xmin;
                        }

                        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;

                glColor3f(1, 0, 0);
                glBegin(GL_LINE_LOOP);
```

```
                glVertex2f(xvmin, yvmin);
                glVertex2f(xvmax, yvmin);
                glVertex2f(xvmax, yvmax);
                glVertex2f(xvmin, yvmax);
                glEnd();

                glColor3f(0, 0, 1);
                glBegin(GL_LINES);
                glVertex2d(vx0, vy0);
                glVertex2d(vx1, vy1);
                glEnd();
        }
}

void display()
{
        glClear(GL_COLOR_BUFFER_BIT);

        glColor3f(0, 0, 1);
        glBegin(GL_LINE_LOOP);
        glVertex2f(xmin, ymin);
        glVertex2f(xmax, ymin);
        glVertex2f(xmax, ymax);
        glVertex2f(xmin, ymax);
        glEnd();
        for (int i = 0; i < n; i++)
        {
                glBegin(GL_LINES);
                glVertex2d(ls[i].x1, ls[i].y1);
                glVertex2d(ls[i].x2, ls[i].y2);
                glEnd();
        }

        for (int i = 0; i < n; i++)
                cohensuther(ls[i].x1, ls[i].y1, ls[i].x2, ls[i].y2);

        glFlush();
}
void myinit()
{
        glClearColor(1, 1, 1, 1);
        glColor3f(1, 0, 0);
        glPointSize(1.0);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        gluOrtho2D(0, 500, 0, 500);
}

void main(int argc, char** argv)
{
        printf("Enter window coordinates (xmin ymin xmax ymax): \n");
        scanf("%lf%lf%lf%lf", &xmin, &ymin, &xmax, &ymax);
        printf("Enter viewport coordinates (xvmin yvmin xvmax yvmax)
:\n");
        scanf("%lf%lf%lf%lf", &xvmin, &yvmin, &xvmax, &yvmax);
        printf("Enter no. of lines:\n");
        scanf("%d", &n);
```

```
        for (int i = 0; i < n; i++)
        {
                printf("Enter line endpoints (x1 y1 x2 y2):\n");
                scanf("%d%d%d%d", &ls[i].x1, &ls[i].y1, &ls[i].x2,
&ls[i].y2);
        }
        glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
        glutInitWindowSize(500, 500);
        glutInitWindowPosition(0, 0);
        glutCreateWindow("clip");
        myinit();
        glutDisplayFunc(display);
        glutMainLoop();
}
```