#include <GL/glut.h>  
#include <vector>  
  
using namespace std;  
  
struct Point {  
    float x, y;  
};  
  
vector<Point> inputPolygon;  
vector<Point> clippedPolygon;  
  
const float xmin = 100, ymin = 100, xmax = 400, ymax = 400;  
bool clipped = false;  
  
void drawPolygon(const vector<Point>& poly, float r, float g, float b) {  
    if (poly.empty()) return;  
    glColor3f(r, g, b);  
    glBegin(GL\_LINE\_LOOP);  
    for (const auto& p : poly)  
        glVertex2f(p.x, p.y);  
    glEnd();  
}  
  
vector<Point> clipEdge(const vector<Point>& poly, char edge) {  
    vector<Point> output;  
    for (int i = 0; i < poly.size(); ++i) {  
        Point curr = poly[i];  
        Point prev = poly[(i + poly.size() - 1) % poly.size()];  
  
        bool currInside, prevInside;  
        float ix, iy;  
  
        switch (edge) {  
            case 'l': currInside = curr.x >= xmin; prevInside = prev.x >= xmin; break;  
            case 'r': currInside = curr.x <= xmax; prevInside = prev.x <= xmax; break;  
            case 'b': currInside = curr.y >= ymin; prevInside = prev.y >= ymin; break;  
            case 't': currInside = curr.y <= ymax; prevInside = prev.y <= ymax; break;  
        }  
  
        if (currInside && prevInside) {  
            output.push\_back(curr);  
        } else if (!prevInside && currInside) {  
            if (edge == 'l' || edge == 'r') {  
                ix = (edge == 'l') ? xmin : xmax;  
                iy = prev.y + (curr.y - prev.y) \* (ix - prev.x) / (curr.x - prev.x);  
            } else {  
                iy = (edge == 'b') ? ymin : ymax;  
                ix = prev.x + (curr.x - prev.x) \* (iy - prev.y) / (curr.y - prev.y);  
            }  
            output.push\_back({ix, iy});  
            output.push\_back(curr);  
        } else if (prevInside && !currInside) {  
            if (edge == 'l' || edge == 'r') {  
                ix = (edge == 'l') ? xmin : xmax;  
                iy = prev.y + (curr.y - prev.y) \* (ix - prev.x) / (curr.x - prev.x);  
            } else {  
                iy = (edge == 'b') ? ymin : ymax;  
                ix = prev.x + (curr.x - prev.x) \* (iy - prev.y) / (curr.y - prev.y);  
            }  
            output.push\_back({ix, iy});  
        }  
    }  
    return output;  
}  
  
void sutherlandHodgman() {  
    clippedPolygon = inputPolygon;  
    for (char edge : {'l', 'r', 'b', 't'}) {  
        clippedPolygon = clipEdge(clippedPolygon, edge);  
    }  
    clipped = true;  
}  
  
void display() {  
    glClear(GL\_COLOR\_BUFFER\_BIT);  
  
    // Draw the clipping window  
    glColor3f(1, 0, 0);  
    glBegin(GL\_LINE\_LOOP);  
    glVertex2f(xmin, ymin);  
    glVertex2f(xmax, ymin);  
    glVertex2f(xmax, ymax);  
    glVertex2f(xmin, ymax);  
    glEnd();  
  
    if (!clipped) {  
        // Show input polygon as it's being drawn (blue)  
        drawPolygon(inputPolygon, 0, 0, 1);  
    } else {  
        // Show clipped polygon only (green)  
        drawPolygon(clippedPolygon, 0, 1, 0);  
    }  
  
    glFlush();  
}  
  
void mouse(int button, int state, int x, int y) {  
    if (state != GLUT\_DOWN) return;  
  
    y = 500 - y;  
  
    if (button == GLUT\_LEFT\_BUTTON && !clipped) {  
        inputPolygon.push\_back({(float)x, (float)y});  
    } else if (button == GLUT\_RIGHT\_BUTTON && !clipped) {  
        sutherlandHodgman();  
    }  
  
    glutPostRedisplay();  
}  
  
void init() {  
    glClearColor(1, 1, 1, 1);  
    gluOrtho2D(0, 500, 0, 500);  
}  
  
int main(int argc, char\*\* argv) {  
    glutInit(&argc, argv);  
    glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);  
    glutInitWindowSize(500, 500);  
    glutCreateWindow("Sutherland-Hodgman Clipping");  
  
    init();  
    glutDisplayFunc(display);  
    glutMouseFunc(mouse);  
  
    glutMainLoop();  
    return 0;  
}