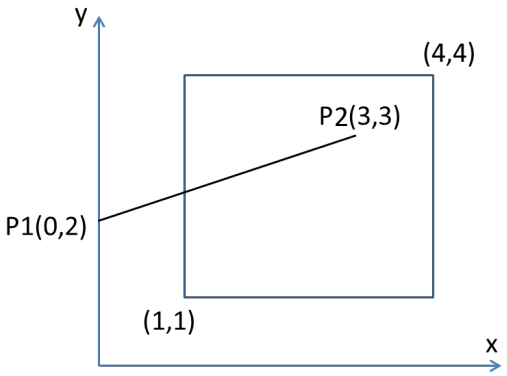
Lab11 for CPT205 Computer Graphics (Clipping)

PART I. Manual Work

1) Cohen-Sutherland 2D Line Clipping

Given a clipping window defined by points (1,1) and (4,4), and a line defined by P1(0,2) and P2(3,3), clip the line using the Cohen-Sutherland 2D line clipping algorithm. You are required to perform the following tasks:

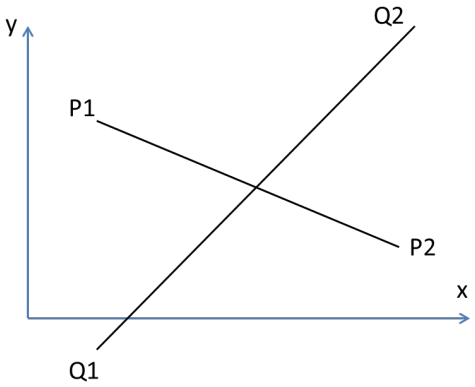


a) Decide the outcodes for the 9 regions of the clipping window plane
b) Decide the outcode for each of the two end points of the line
c) Provide the main steps of your working
d) Show the final results for rendering the line after clipping

2) Parametric Lines

For two line segments in a plane, represented in parametric form:

$\mathbf{p}(\alpha)=(1-\alpha)\mathbf{p}1+\alpha\mathbf{p}2$
$\mathbf{q}(\beta)=(1-\beta)\mathbf{q}1+\beta\mathbf{q}2$



a) propose a procedure for determining whether the segments intersect and,
b) if so, propose a procedure for finding the point of intersection.

PART II. Implementation of Cohen-Sutherland 2D Line Clipping

1) Sample program for Cohen-Sutherland 2D Line Clipping

This program is written with OpenGL to display the 9 regions and a line segment for Cohen-Sutherland 2D Line Clipping.

```
// Lab11: OpenGL implementation of 9 regions and a line segment for Cohen-Sutherland 2D Line Clipping

#define FREEGLUT_STATIC
#include <GL/freeglut.h>
#include"windows.h"

#define MAX_CHAR 128

typedef struct { GLfloat x, y; } point;
point P1 = { 130,270 }, P2 = { 350,350 };

GLfloat win_x = 600, win_y = 600;
GLfloat win_x_left = win_x / 3, win_x_right = win_x / 3 * 2, win_y_bottom = win_y / 3, win_y_top = win_y
/ 3 * 2;

void drawString(const char* str) {  // Display characters for the outcodes
    static int isFirstCall = 1;
    static GLuint lists;
    if (isFirstCall) {
        isFirstCall = 0;
        lists = glGenLists(MAX_CHAR);  // Generate a contiguous set of empty display lists
        wglUseFontBitmaps(wglGetCurrentDC(), 0, MAX_CHAR, lists);  // Convert characters into images
    }
    for (; *str != '\0'; ++str) {
        glCallList(lists + *str);  // Draw bitmap characters
    }
}

void selectFont(int size, int charset, const char* face) {  // Select a font face and size
    HFONT hFont = CreateFontA(size, 0, 0, 0, FW_MEDIUM, 0, 0, 0, charset, OUT_DEFAULT_PRECIS,
CLIP_DEFAULT_PRECIS, DEFAULT_QUALITY, DEFAULT_PITCH | FF_SWISS, face);
    HFONT hOldFont = (HFONT)SelectObject(wglGetCurrentDC(), hFont);
    DeleteObject(hOldFont);
}

void PlotLine(GLfloat x1, GLfloat y1, GLfloat x2, GLfloat y2) {  // Draw a solid straight line
    glBegin(GL_LINES);
    glVertex2f(x1, y1);
    glVertex2f(x2, y2);
    glEnd();
}

void renderScene(void) {  // Draw outcode regions, outcode values, and a line segment to be clipped
    glClearColor(0, 0, 0, 0);
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(1, 1, 1);

    selectFont(24, ANSI_CHARSET, "Comic Sans MS");
    //plot outcodes
    glRasterPos2f(win_x_left / 2 - 30, win_y_top / 2 + win_y / 2);
    drawString("1001");
    glRasterPos2f(win_x_left / 2 - 30, win_y_top / 2 + win_y_bottom / 2);
    drawString("0001");
    glRasterPos2f(win_x_left / 2 - 30, win_y_bottom / 2);
    drawString("0101");
    glRasterPos2f(win_x_right / 2 - 30 + win_x_left / 2, win_y_top / 2 + win_y / 2);
    drawString("1000");
    glRasterPos2f(win_x_right / 2 - 20 + win_x_left / 2, win_y_top / 2 + win_y_bottom / 2);
    drawString("0000");
    glRasterPos2f(win_x_right / 2 - 30 + win_x_left / 2, win_y_bottom / 2);
    drawString("0100");
    glRasterPos2f(win_x_right / 2 - 30 + win_x / 2, win_y_top / 2 + win_y / 2);
    drawString("1010");
    glRasterPos2f(win_x_right / 2 - 30 + win_x / 2, win_y_top / 2 + win_y_bottom / 2);
    drawString("0010");
    glRasterPos2f(win_x_right / 2 - 30 + win_x / 2, win_y_bottom / 2);
    drawString("0110");

    glLineWidth(1);
```

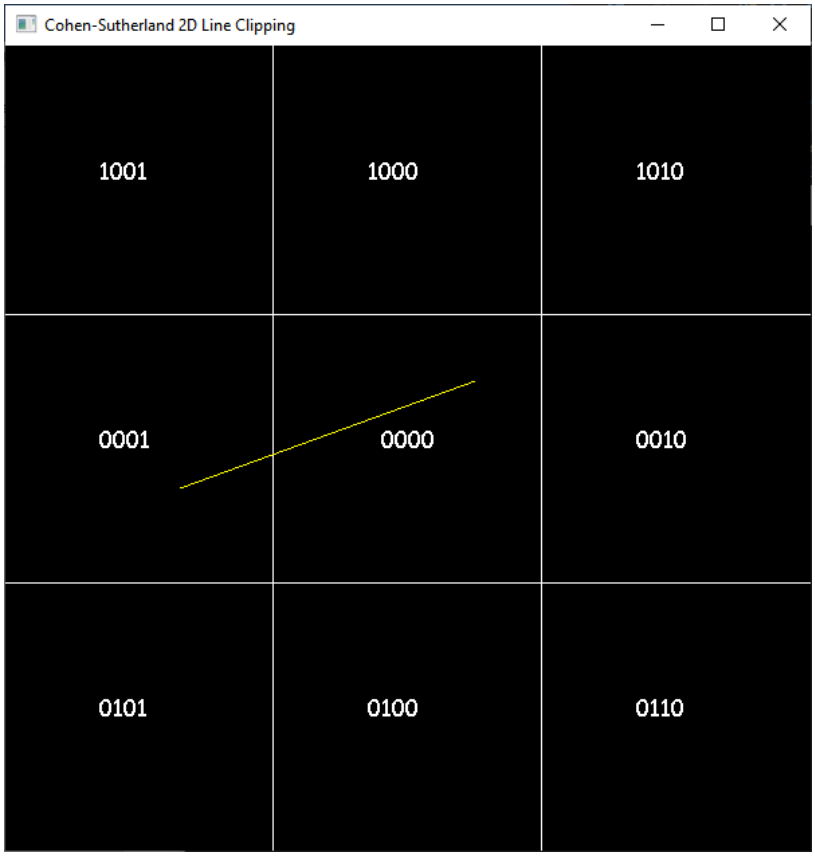
```
    PlotLine(win_x_left, 0, win_x_left, win_y);
    PlotLine(0, win_y_top, win_x, win_y_top);
    PlotLine(win_x_right, 0, win_x_right, win_y);
    PlotLine(0, win_y_bottom, win_x, win_y_bottom);

    glColor3f(1, 1, 0);
    PlotLine(P1.x, P1.y, P2.x, P2.y);  // Draw original line segment

    glutSwapBuffers();
}

void myinit(void) {  // Initialisation
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0, win_x, 0, win_y);
}

int main(int argc, char* argv[]) {  // The main program
    glutInit(&argc, (char**)argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
    glutInitWindowPosition(100, 100);
    glutInitWindowSize(win_x, win_y);
    glutCreateWindow("Cohen-Sutherland 2D Line Clipping");
    myinit();
    glutDisplayFunc(renderScene);
    glutMainLoop();
    return 0;
}
```



2) Further Work

Extend the sample program so that

a) the regions outside the clipping window are shown in dashed lines, and
b) the line is clipped and the portion outside the clipping window is shown in dashed line.