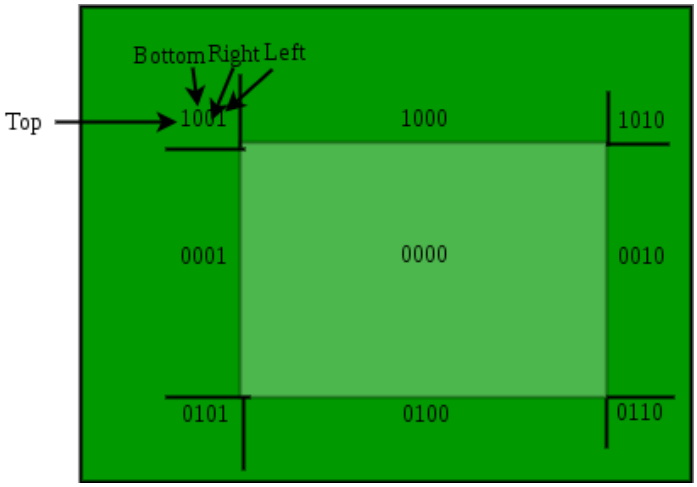
1.Line Clipping (CohenSutherland Algorithm)

Cohen-Sutherland algorithm divides a two-dimensional space into 9 regions and then efficiently determines the lines and portions of lines that are inside the given rectangular area.
The algorithm can be outlines as follows:-
For a given line extreme point (x, y), we can quickly find its region's four bit code. Four bit code can be computed by comparing x and y with four values (xmin, xmax, ymin and ymax).

1. If x is less than xmin then bit number 1 is set.
2. If x is greater than xmax then bit number 2 is set.
3. If y is less than ymin then bit number 3 is set.
4. If y is greater than ymax then bit number 4 is set

Code for finding area:



2. Results

OpenGL code

```
   #ifdef WIN32
#include <windows.h>
#endif

#include <stdlib.h>
#include <iostream>
#include <fstream>

#ifdef __APPLE__
#include <GLUT/glut.h>
#include <OpenGL/gl.h>
#include <OpenGL/glu.h>
#else
#include <GL/glut.h>
#include <GL/glu.h>
```

```
#include <GL/gl.h>
#endif
#include <OpenGL/gl3.h>
#include <stdio.h>
#include <math.h>

using namespace std;

int minvalueX, minvalueY, maxvalueX, maxvalueY;
int firstvalueX, firstvalueY, secondvalueX, secondvalueY;

int code1[4] = { 0, 0, 0, 0 };
int code2[4] = { 0, 0, 0, 0 };

bool isReject = false;
int getcode2(int x,int y);
int getcode1(int x ,int y);
void generateCodeForPoints()
{
    getcode1(firstvalueX, firstvalueY);
    getcode2(secondvalueX, secondvalueY);
}


void drawLine(int x0, int y0, int x1, int y1)
{
    glBegin(GL_LINES);
    glVertex2i(x0, y0);
    glVertex2i(x1, y1);
    glEnd();
}

void SutherlandHodgemanPolygonClipping();
void myDisplay(void);

int main(int argc, char** argv)
{

    cout << "Enter minimum window co-ordinates( x and y co-ordinates) : "; cin >> minvalueX >>
    cout << "Enter maximum window co-ordinates ( x and y co-ordinates) : "; cin >> maxvalueX >>

    cout << "Enter x and y value of first point of line: "; cin >> firstvalueX >> firstvalueY;
    cout << "Enter x and y value of second point of line: "; cin >> secondvalueY >> secondvalueX

    // 4 digit code generator
    generateCodeForPoints();
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(640, 480);
    glutInitWindowPosition(100, 150);
    glutCreateWindow("Clipping with Cohen-Sutherland algorithm");
    glutDisplayFunc(myDisplay);



    glClearColor(1.0, 1.0, 1.0, 0.0);
```

```
    glColor3f(0.0f, 0.0f, 0.0f);
    glPointSize(4.0);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0.0, 640.0, 0.0, 480.0);

    glutMainLoop();

}

void myDisplay(void)
{
    glClear(GL_COLOR_BUFFER_BIT);

    glColor3f(1.0, 0.0, 1.0);
    glRecti(minvalueX, minvalueY, maxvalueX, maxvalueY);

    glColor3f(0.0, 0.0, 0.0);
    SutherlandHodgemanPolygonClipping();

    glFlush();
}

int getcode1(int x, int y) {
    if (y > maxvalueY)
    {
        code1[0] = 1;
    }

    if (y < minvalueY)
    {
        code1[1] = 1;
    }

    if (x > maxvalueX)
    {
        code1[2] = 1;
    }

    if (x < minvalueX)
    {
        code1[3] = 1;
    }

    int codeRes1 = code1[0] * 1000 + code1[1] * 100 + code1[2] * 10 + code1[3];

    return codeRes1;
}

void SutherlandHodgemanPolygonClipping()
{
    // Is both point accepted
    if (getcode1(firstvalueX, firstvalueY) == 0 && getcode2(secondvalueX, secondvalueY) == 0 &&
    {
        drawLine(firstvalueX, firstvalueY, secondvalueX, secondvalueY);
    }
```

```
else
{
    for (int i = 0; i < 4; i++)
    {
        // Is both point rejected
        if (code1[i] == code2[i] && code1[i] == 1)
        {
            isReject = true;
            break;
        }


    }

    if (isReject)
    {
        cout << "Both point rejected" << endl;
    }
    else
    {
        //cout << "Not rejected" << endl;

        for (int i = 0; i < 4; i++)
        {
            if (code1[i] == 1)
            {
                switch (i)
                {

                    case 0:
                        firstvalueY = maxvalueY;
                        break;

                    case 1:
                        firstvalueY = minvalueY;
                        break;

                    case 2:
                        firstvalueX = maxvalueX;
                        break;

                    case 3:
                        firstvalueX = minvalueX;
                        break;
                }
            }

            if (code2[i] == 1)
            {
                switch (i)
                {

                    case 0:
                        secondvalueY = maxvalueY;
                        break;
```

```
                        case 1:
                            secondvalueY = minvalueY;
                            break;

                        case 2:
                            secondvalueX = maxvalueX;
                            break;

                        case 3:
                            secondvalueX = minvalueX;
                            break;
                    }
                }



            }

            drawLine(firstvalueX, firstvalueY, secondvalueX, secondvalueY);
        }
    }
}

int getcode2(int x, int y) {
    if (y > maxvalueY)
    {
        code2[0] = 1;
    }

    if (y < minvalueY)
    {
        code2[1] = 1;
    }

    if (x > maxvalueX)
    {
        code2[2] = 1;
    }

    if (x < minvalueX)
    {
        code2[3] = 1;
    }

    int codeRes2 = code2[0] * 1000 + code2[1] * 100 + code2[2] * 10 + code2[3];

    return codeRes2;
}
```

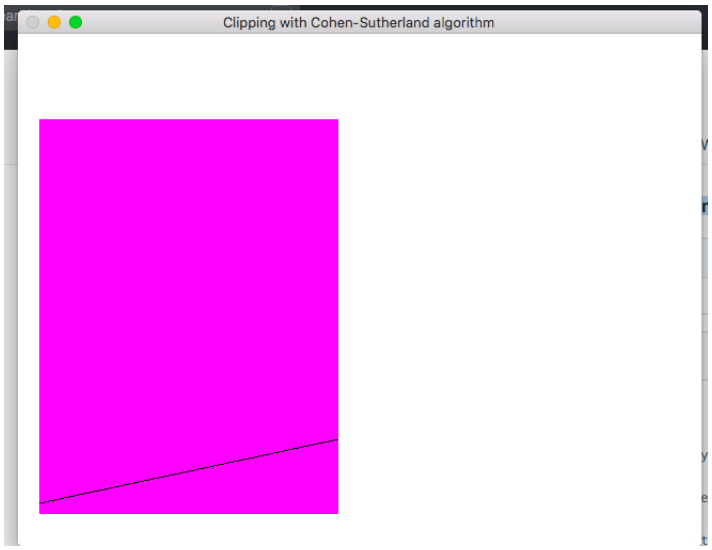
3. Scale

input like:
Enter minimum window co-ordinates(x and y co-ordinates) : 60 70
Enter maximum window co-ordinates (x and y co-ordinates) : 300 200
Enter x and y value of first point of line: 20 40

Graphics Assignment No:- 7

Enter x and y value of second point of line: 300 400
Program ended with exit code: 9

Result 1:



Result 2: