**Title: I**mplementation of Line Clipping Algorithm**.**

#### **Problem Statement:** Implement Cohen Sutherland line clipping method to clip the line with respect to the view-port and window. Use mouse click, keyboard interface.

**Objective;**

i) To study what the clipping is?

ii) To study and implement the Cohen Sutherland line clipping algorithm.

**Theory:**

**Line Clipping – Cohen Sutherland**

In computer graphics, '*line clipping'* is the process of removing lines or portions of lines outside of an area of interest. Typically, any line or part thereof which is outside of the viewing area is removed.

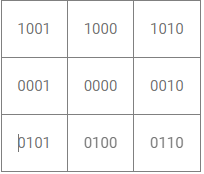
The Cohen–Sutherland algorithm is a computer graphics algorithm used for line clipping. The algorithm divides a two-dimensional space into 9 regions (or a three-dimensional space into 27 regions), and then efficiently determines the lines and portions of lines that are visible in the center region of interest (the viewport).

The algorithm was developed in 1967 during flight simulator work by Danny Cohen and Ivan Sutherland

The design stage includes, excludes or partially includes the line based on where:

* Both endpoints are in the viewport region (bitwise OR of endpoints == 0): trivial accept.
* Both endpoints share at least one non-visible region which implies that the line does not cross the visible region. (bitwise AND of endpoints != 0): trivial reject.
* Both endpoints are in different regions: In case of this nontrivial situation the algorithm finds one of the two points that is outside the viewport region (there will be at least one point outside). The intersection of the outpoint and extended viewport border is then calculated (i.e. with the parametric equation for the line) and this new point replaces the outpoint. The algorithm repeats until a trivial accept or reject occurs.

The numbers in the figure below are called outcodes. The outcode is computed for each of the two points in the line. The outcode will have four bits for two-dimensional clipping, or six bits in the three-dimensional case. The first bit is set to 1 if the point is above the viewport. The bits in the 2D outcode represent: Top, Bottom, Right, Left. For example the outcode 1010 represents a point that is top-right of the viewport. Note that the outcodes for endpoints must be recalculated on each iteration after the clipping occurs.



## **Algorithm**

**Steps**

1) Assign the region codes to both endpoints.

2) Perform OR operation on both of these endpoints.

3) if OR = 0000,

then it is completely visible (inside the window).

else

Perform AND operation on both these endpoints.

i) if AND != 0000,

then the line is invisible and not inside the window. Also, it **can’t be** considered for clipping.

ii) else

AND = 0000, the line is partially inside the window and **considered for clipping.**

4) After confirming that the **line is partially inside** the window, then we find the intersection with the boundary of the window. By using the following formula:-

Slope:- m= (y2-y1)/(x2-x1)

a) If the line passes through the top or the line intersects with the top boundary of the window.

**x = x + (y\_wmax – y)/m**

**y = y\_wmax**

b) If the line passes through the **bottom** or the line intersects with the bottom boundary of the window.

**x = x + (y\_wmin – y)/m**

**y = y\_wmin**

c) If the line passes through the **left** region or the line intersects with the left boundary of the window.

**y = y+ (x\_wmin – x)\*m**

**x = x\_wmin**

d) If the line passes through the **right** region or the line intersects with the right boundary of the window.

**y = y + (x\_wmax -x)\*m**

**x = x\_wmax**

5) Now, overwrite the endpoints with a new one and update it.

6) Repeat the 4th step till your line doesn’t get completely clipped

Given a set of lines and a rectangular area of interest, the task is to remove lines that are outside the area of interest and clip the lines which are partially inside the area.

Input : Rectangular area of interest (Defined by

below four values which are coordinates of

bottom left and top right)

x\_min = 4, y\_min = 4, x\_max = 10, y\_max = 8

A set of lines (Defined by two corner coordinates)

line 1 : x1 = 5, y1 = 5, x2 = 7, y2 = 7

Line 2 : x1 = 7, y1 = 9, x2 = 11, y2 = 4

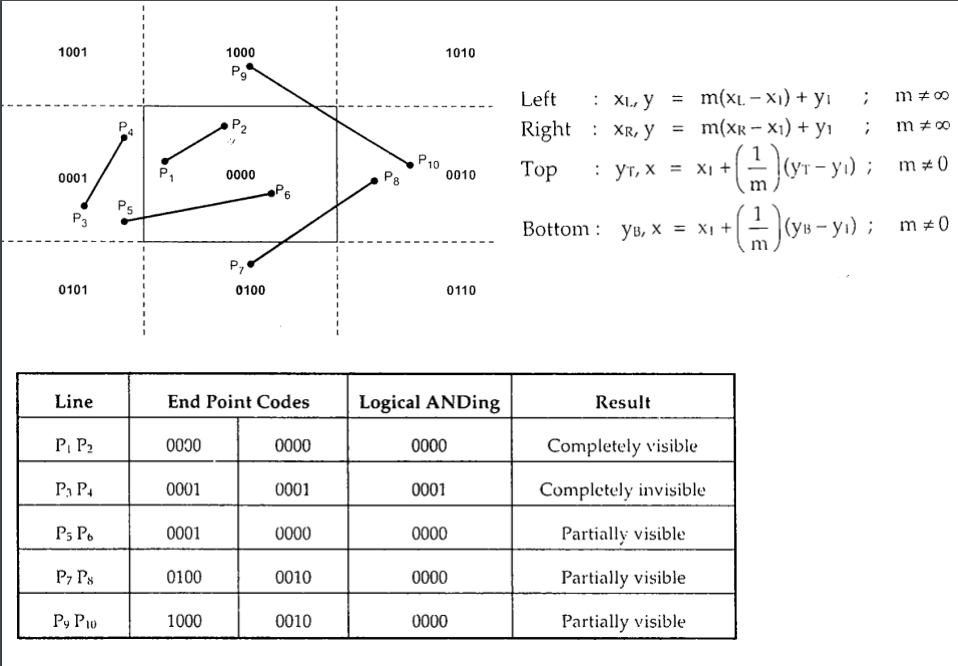
Line 3 : x1 = 1, y1 = 5, x2 = 4, y2 = 1

Output : Line 1 : Accepted from (5, 5) to (7, 7)

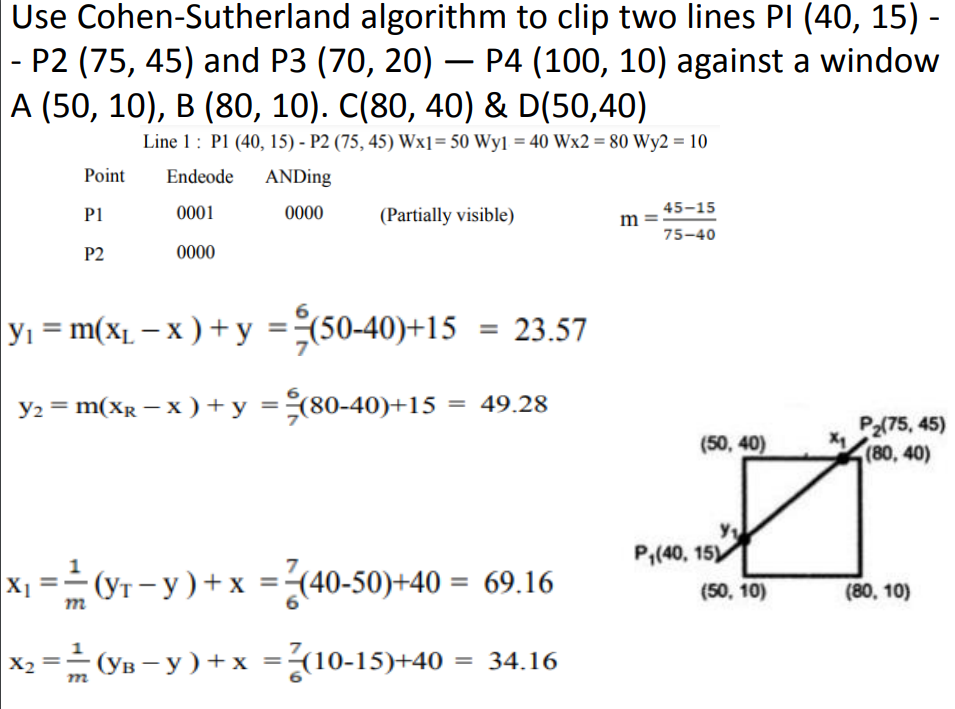
Line 2 : Accepted from (7.8, 8) to (10, 5.25)

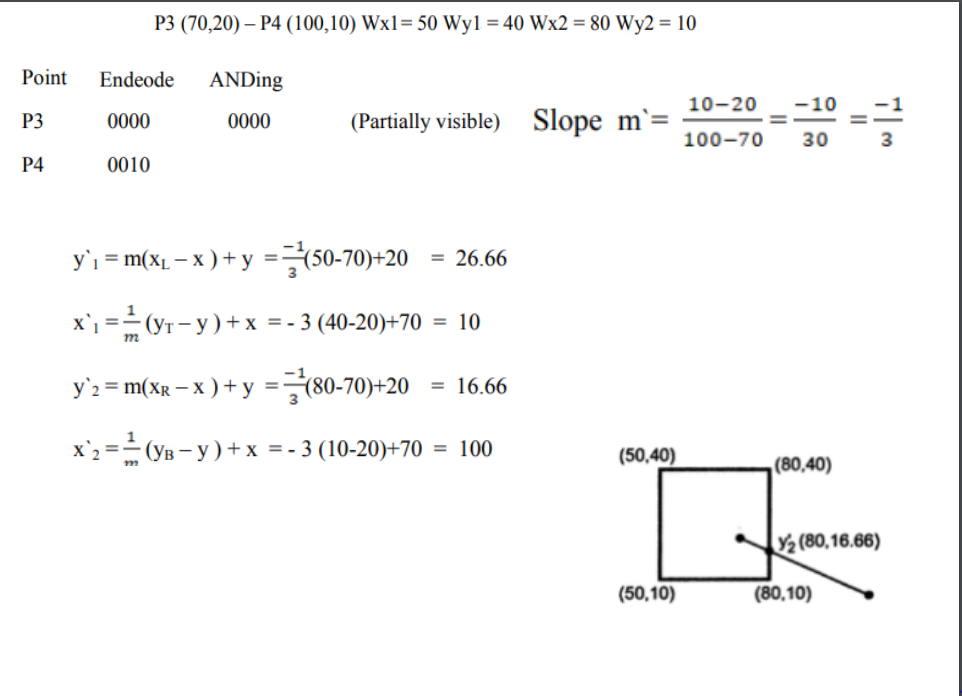
Line 3 : Rejected

**Summary of Cohen Sutherland Line Clipping Algorithm;**

****

**Example:**

****

****

**Conclusion: We have studied and implemented the Cohen Sutherland Line Clipping Algorithm.**