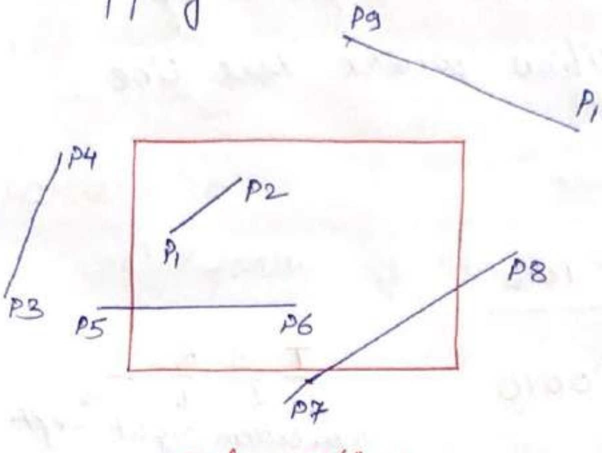
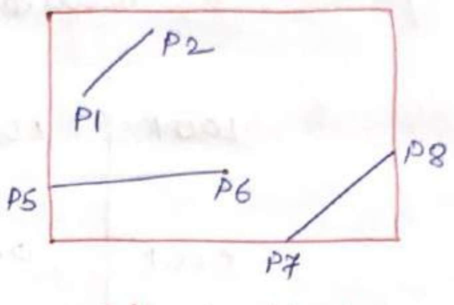
Line clipping:

- Line clipping involves several possible cases:
  1. Completely inside the clipping window.
  2. Completely outside the clipping window.
  3. Partially inside and partially outside the clipping window.



Before clipping



After clipping.

- Line which is completely inside is display completely. Line which is completely outside is eliminated from display. And for partially inside line we need to calculate intersection with window boundary and find which part is inside the clipping boundary and which part is eliminated.

- For line clipping several methods were proposed to solve clipping procedure. Some of them are discusse

Cohen - Sutherland Line Clipping :

- this is one of the oldest and most popular line clipping procedures.
- In this we devide whole space into nine region and assign 4 bit code to each endpoint of line depending on the position where the line endpoint is located.

1001	1000	1010
0001	0000	0010
0101	0100	0110

T B R L

Top bottom right Left

- Above figure shows code for line end point which is fall within particular area.
- Code is deriving by setting particular bit according to position of area.

set bit Left : for left side clipping window

set bit R : for right side clipping window

set bit B : for bottom side clipping window

set bit T : for top side clipping window.

- All bits as mention above are set means 1 and other are 0.

Algorithm :-

```
step 1:
1.1 Assign region code to both endpoint of a line
    depending on the position where the line endpoint
    is located.

step 2:
2.1 If both endpoints have code '0000'
    2.1.1 Then line is completely inside.
2.2 otherwise
    2.2.1 Perform logical ending between this two codes.
    2.2.2 If result of logical ending is non zero
        2.2.2.1 Line is completely outside the clipping
                window.
    2.2.3 otherwise
        2.2.3.1 Calculate the intersection point with the
                boundary one by one
        2.2.3.2 Divide the line into two parts from
                intersection point
        2.2.3.3 Recursively call algorithm for both line
                segments.

step 3:
3.1 Draw line segment which are completely inside
    and eliminate other line segment which found
    completely outside.
```

Intersection point calculations with clipping window boundary.

- For intersection calculation we use line equation $y = mx + b$.

- 'x' is constant for left & right boundary which is:
  - for left $x = x_{wmin}$
  - for right $x = x_{wmax}$

- So we calculate y coordinates of intersection for this boundary as

$$y = y_1 + m(x - x_1)$$

value of x depends on boundary is left or right.

- 'y' is constant for top & bottom boundary which is:
  - for top $y = y_{wmax}$
  - for bottom $y = y_{wmin}$.

- So we calculate y coordinate of intersection for this boundary as:

$$x = x_1 + \frac{y - y_1}{m}$$

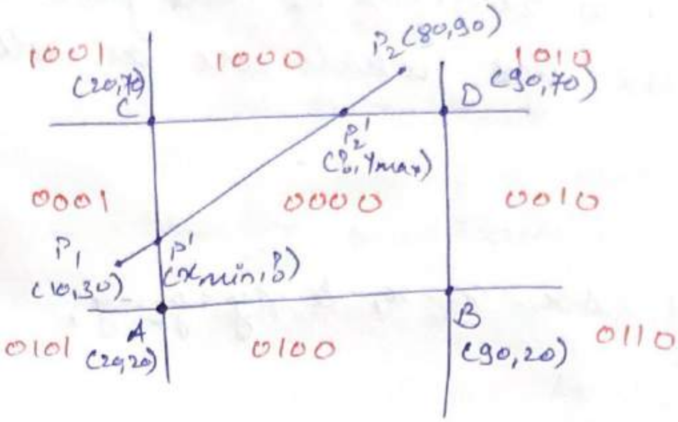
value of y depends on boundary is top or bottom.

Example :

Q1. $(x_{min}, y_{min}) = (20, 20)$
$(x_{max}, y_{max}) = (90, 70)$ } window co ordinate

Line end points are given as $P_1 (10, 30)$ & $P_2 (80, 90)$. clip the line against window.

sol:ⁿ



code of P_1 = 0001
code of $P2$ = 1000
AND 0000

Since AND of codes of P1 & P2 is 0. Hence line partially lies inside clipping window.

$$m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{90 - 30}{80 - 10} = \frac{60}{70} = \frac{6}{7} = 0.857$$

$$y = m(x_{min} - x_1) + y_1$$
$$= 0.857(20 - 10) + 30$$
$$= 38.57$$

$$x = 10 + \frac{70 - 30}{0.857}$$
$$= 56.67$$

clipped coordinates are $P_1' = (20, 38.57)$
$P_2' = (56.67, 70)$