1. Implement Cohen-Sutherland Line Clipping Algorithm.

***Algorithm:***

1. Assign a region code for each end-points.
2. If both end-points have **region code 0000** → ***trivially accept*** these lines.
3. Else, perform the logical **AND** operation for both region codes.  
   If the result is **NOT** **0000** → ***trivially reject*** the line.  
   Else (i.e., **result = 0000**, need clipping)
   1. Choose an end-point of the line that is outside the window.
   2. Find the intersection point at the window boundary (based on region code).
   3. Replace end-point with the intersection point and update the region code.
   4. Repeat step (ii) until we find a clipped line either trivially accepted or trivially rejected.
4. Repeat step (i) for other lines.

**Intersection Calculations**

Intersection with vertical boundary:

Intersection with horizontal boundary:

1. Implement Liang-Barsky Line Clipping Algorithm.

***Algorithm:***

1. Given, end points of the line *(x1, y1)* and *(x2, y2).*
2. Calculate the value of *pk* and *qk* for *k = 1, 2, 3, 4* as:
   1. For *k = 1*, checking whether the line is inside left boundary:
   2. For *k = 2*, checking whether the line is inside right boundary:
   3. For *k = 3*, checking whether the line is inside bottom boundary:
   4. For *k = 4*, checking whether the line is inside top boundary:
3. If *pk = 0* and *qk < 0,* the line is completely outside and is rejected.  
   Else, go to step (iv).
4. If *pk < 0,* the line proceeds from outside to inside and we calculate *u1* as:  
    If *pk > 0,* the line proceeds from inside to outside and we calculate *u2* as:
5. If *u1 > u2,* then the line is outside and is rejected.  
   Else, go to step (vi).
6. The points are calculated as:  
    ,   
   and, ,