Cohen-Sutherland Line Clipping

In this problem, you are given a clipping window (defining the "viewport") and an example line, and asked to perform the steps of the line clipping algorithm we just learned in class. Assume the origin is at the top left and y is increasing going down (like HTML canvas).

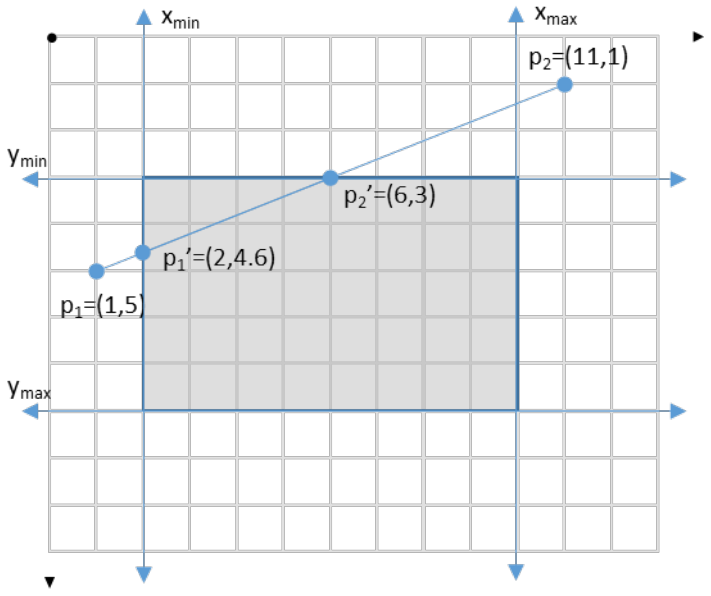
Input: viewport defined by the lines $x_{min} = 2$, $x_{max} = 10$, $y_{min} = 3$, $y_{max} = 8$.
line defined by the points $p_1 = (1, 5)$ and $p_2 = (11, 1)$.

Output: p_1' and p_2', the points defining the line that should actually be drawn.

1. Draw out the viewport and the example line, labeling p_1, p_2, x_{min}, x_{max}, y_{min}, and y_{max}.
2. Write out the binary 4-digit codes for p_1 and p_2.

   p_1 : 0001
   p_2 : 1010

3. Write out what case each point falls under and show how the algorithm would update the points. What are the final p_1' and p_2'?



p_1' : case 3d. New point is y = -0.4(2-1)+5 = 4.6, x = 2.
p_2': case 3e. New point is x = (3-1)/(-0.4)+11 = 6, y = 3

4. Label p_1' and p_2' on your picture and make sure they agree visually with your calculations.
5. How many "rounds" of clipping are required to make this example line within the viewport? *2*

Pseudocode for Cohen-Sutherland:

```
cohen_sutherland(p1,p2):
  code1 = getCode(p1);
  code2 = getCode(p2);
  if (code1|code2 == 0000) then                                  // case 1
    drawLine(p1,p2)
  else if (code1 & code2 != 0000) then                           // case 2
    // skip line
  else if (code1 & 1000) then                                    // case 3a
    p1new.x ← (ymin-p1.y)/m+p1.x, p1new.y ← ymin,
    cohen_sutherland(p1new,p2)
  else if (code1 & 0100) then                                    // case 3b
    p1new.x ← (ymax-p1.y)/m+p1.x, p1new.y ← ymax,
    cohen_sutherland(p1new,p2)
  else if (code1 & 0010) then                                    // case 3c
    p1new.y ← m(xmax-p1.x) +p1.y, p1new.x ← xmax,
    cohen_sutherland(p1new,p2)
  else if (code1 & 0001) then                                    // case 3d
    p1new.y ← m(xmin-p1.x)+p1.y, p1new.x ← xmin,
    cohen_sutherland(p1new,p2)
  else if (code2 & 1000) then                                    // case 3e
    p2new.x ← (ymin-p2.y)/m+p2.x, p2new.y ← ymin,
    cohen_sutherland(p1,p2new)
  else if (code2 & 0100) then                                    // case 3f
    p2new.x ← (ymax-p2.y)/m+p2.x, p2new.y ← ymax,
    cohen_sutherland(p1,p2new)
  else if (code2 & 0010) then                                    // case 3g
    p2new.y ← m(xmax-p2.x) +p2.y, p2new.x ← xmax,
    cohen_sutherland(p1,p2new)
  else if (code2 & 0001) then                                    // case 3h
    p2new.y ← m(xmin-p2.x)+p2.y, p2new.x ← xmin,
    cohen_sutherland(p1,p2new)
```

6. Why are there eight subparts to case 3? What does each one represent/do?

   Each compares one of the endpoints with one of the viewport boundaries. 2 points × 4 sides = 8 cases.

7. A four-bit sequence allows for sixteen possible values, yet we only have nine regions. Which bit sequences are not used, and why do they represent nonsensical situations?

The unused values are nonsensical because they imply that a point is simultaneously both above and below the viewport, etc.

8. Which case would be activated for each of the following pairs of codes?
   a. 1010 and 0101 *3a*
   b. 0000 and 1001 *3e*
   c. 0110 and 0101 *2*
   d. 0001 and 0001 *2*