Graphics Lab Exercise Date 07-02-2019

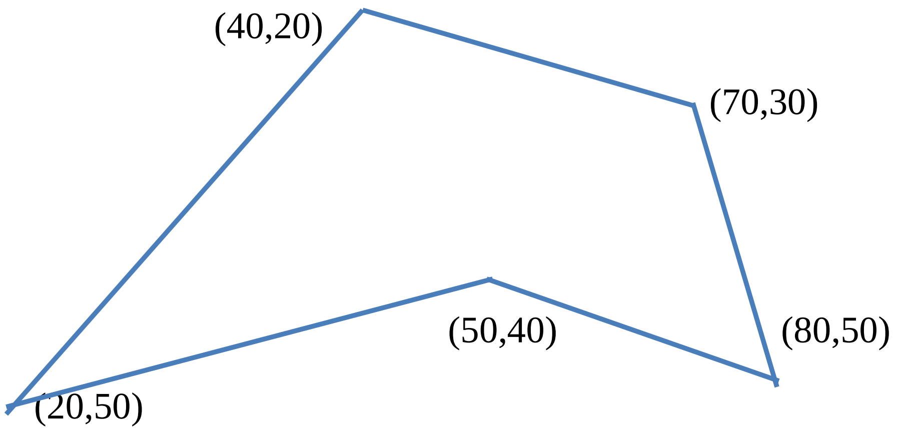
1.Write a C program to implement Liang and Barsky line Clipping Algorithm against a rectangular clip window.

2.Write a C program to implement Cohen Sutherland line Clipping Algorithm against a rectangular clip window.

Implementation should be tested for various cases using the lines between P_1 and p*2* as shown in Table below. The rectangular clip window coordinates are $x_{min=40}$, $x_{max=100}$, $y_{min=40}$, $y_{max=80}$

Line Number	P_1	P_2
Line 1	(30,65)	(55,30)
Line2	(60,20)	(110,90)
Line 3	(60,100)	(80,70)
Line 4	(85,50)	(120,75)

3. .Write a C program to implement Flood fill Algorithm for a polygon.



Instructions

1. Steps of Liang and Barsky line Clipping Algorithm:

Input: A line segments with end points $P(x_1,y_1))$ and $Q(x_2,y_2)$, the window parameters (x_{min}, x_{max}, y_{min}, y_{max}). A window boundary is denoted by *k* where *k* can take the values 1, 2, 3 or 4 corresponding to left, right, below and above boundary, respectively.
Output: clipped line segment.

```
1.  Calculate $\Delta x=x_2-x_1$ and $\Delta y=y_2-y_1$
2.  Calculate $p_1=-\Delta x$, $q_1=x_1-x_{min}$
3.  Calculate $p_2=\Delta x$, $q_2=xmax-x_1$
4.  Calculate $p_3=-\Delta y$, $q_3=y_1-y_{min}$
5.  Calculate $p_4=\Delta y$, $q_4=y_{max}-y_1$
6.  if $p_k=0$ and $q_k<0$ for any k=1,2,3,4 then
            Discard the lines as it is completely outside the window
7.  else
            Compute $r_k=q_k/p_k$ for all those boundaries for which $p_k<0$. Determine
            parameter $u_1=\max\{0,r_k\}$.
            Compute $r_k=q_k/p_k$ for all those boundaries for which $p_k>0$. Determine
            parameter $u_2=\min\{1,r_k\}$.
            if $u_1>u_2$ then
                Eliminate the line as it is completely outside the window
            else if $u_1=0$ then
                There is one intersection point, calculate as $x_2=x_1+u_2\Delta x=y1+u_2\Delta y$
                Return the two end points $(x_1, y_1)$ and $(x_2,y_2)$
            else
                There are two intersection points, calculate as: $x_{1'}=x_1+u_1\Delta$,
                $y_{1'}=y_1+u_1\Delta y$ and $x_2=-x_1+u_2\Delta x$, $y_2=y_1+u_2\Delta$
                Return the two end points $(x_{1'}, y_{1'})$ and $(x_2, y_2)$.
 9.         end if
 10. end if
```

2. Steps of Cohen Sutherland line Clipping Algorithm:

Input: A line segment wih end points PQ and the window parameters (x_{min}, x_{max}, y_{min}, y_{max})
Output: Clipped line segment (NULL if line is completely outside)

```
1. for each end point with coordinate (x, y), where sign(a)=1 if a is positive, 0 otherwise do
        Bit 3 = $\text{sign}(y-y_{max})$
        Bit 2 = $\text{sign}(y_{min}-y)$
        Bit 1 = $\text{sign}(x-x_{max})$
        Bit 0 = $\text{sign}(x_{min}-x)$
end for
2.if both end point region points are 000 then
        return PQ.
3.else if logical AND (i.e., bitwise AND) of the point region code $\neq$ 000 then
        Return NULL
```

```
4.else
    for each boundary $b_i$ where $b_i$ = above, below, right, left, do
        Check corresponding bit bit values of the two end point region codes
        If the bit values are same, then
            Check the next boundary
        else
            Determine $b_i$-line intersection point using line equation
            Assign region code to the intersection point
            Discard the line from the end point outside $b_i$ to the intersection point
            (as it is outside the window)
            If the region codes of both the intersection point and the remaining end
            point are 0000 then
                Reset PQ with the new end points
            end if
        end if
    end for
5.Return modified PQ
6.End if
```

3. Steps of Flood fill Algorithm:

Input: Interior pixel color, specified color, and the seed (interior pixel) *p*

Output: Interior pixels with secified color

```
1.Push (p) to Stack
2.repeat
3.  Set current pixel =Pop(Stack)
4.  Apply specified color to the current pixel
5.  for Each of the four connected pixels (four-connected) or eight connected pixels
        (eight-connected) of current pixel do
6.      If (Color(connected pixel)=interior color then
            Push(connected pixel)
        end if
    end for
7.Until stack empty
```