Ahmet Faruk Ulutaş - 21803717 – CS478 – HW1

1- In the while loop, lines are created between P and all points in the linked list. While navigating the entire linked list, the cross product difference of the two lines (the one above and the next) is added to the sum value at each step.

If the sum value is positive after the loop ends, the list is in counterclockwise order according to P.

```
procedure IsListCounterclockwiseOrderToP( HEAD, P):

begin

    sum = 0

    start = head.next

    ptr = head.next

    finished = FALSE

    while    (finished = TRUE)

            sum += (P.x - ptr.x) * (P.y - ptr.next.y) - (P.y - ptr.y) * (P.x - ptr.next.x)

            if        (ptr.next = START)

                    finished = TRUE

            endif

            ptr = ptr.next

    endwhile

    if        ( sum > 0)

            return TRUE

    endif

    return FALSE

end
```

2- With the naive approach, each face is traversed through a for loop. The faces where the current face intersects are saved in a multidimensional list.

It is checked whether each face in the list intersects with each other. If any of the intersection faces intersect, it cannot be painted using 2 colors.

```
procedure IsFacesPaintableWithTwoColors( DCEL):

begin

        for each face in DCEL /* found via edges or directly in DCEL */

        Add faces that intersect with this face to the list /*like f1 n f2,f3,f4 */

        endfor

        for each face in the list as f /* list[f][0] */

        for each face’ intersections in the list as i /* from 1 to end of the row list[f][i] */

                for each selected face 'i', traverse the i'th row as j

                        if        (list[f][i] == list[i][j])

                                  return FALSE

                        endif

                        endfor

                endfor

        endfor

        return TRUE

end
```

3- With the sweeping line algorithm, they are ordered from left to right along all x axes. Then the intersections with the vertical lines are found. Then we check the intersection numbers in BST one by one. We record the maximum number of intersections in count. Then we multiply the count with the fixed height number and report it.

```
procedure CalMinLengthOfTower( Points[0..2n-1]):

Sort Points left to right along all x axes.

Create empty BST.

for     (i = 0 to 2n-1)  /* This section can be thought of as a method for the sweeping line algorithm.
*/

        if      (point is on the left)

                insert line to tree

        endif

        else

                delete line from tree

        endelse

endfor

/* This section can be thought of as a method for the sweeping line algorithm. */

for each line in the bst tree

        if      (newCount > count)

                count = newCount

        endif

endfor

report count * constant_unit_hegiht
```

4- Traverse the edges of the given face in the given graph.

Delete every edge that is not the edge of the outer component.

Then check both corners of all edges.

If a corner is not an outer component and does not match any corner other than itself, delete that edge. Continue until there are no edges left in this state.

```
procedure expandFace( DCEL, face):

Determine the half-edge of face as edge

start = face.edge

prev = face.edge

while    (next(prev) != start)

         if        (edge.twin != NULL)

                   edge = edge.next

                   DeleteEdge(edge.prev)

         endif

         prev = edge

endwhile

halfEdge = dcel.origin

list = [halfEdge]

twin = halfEdge.twin.next

while    (halfEdge != twin)

         list.append(twin)

         twin = twin.twin.next

Endwhile

v1Check = FALSE, v2Check = FALSE

for each edge in list

         for each otherEdge in list

                   if        ((edge != otherEdge) and (edge.vertex == otherEdge.vertex) or

                            (edge.vertex == otherEdge.otherVertex)
```

```
            v1Check = TRUE
        endif
        if
            ((edge != otherEdge) and (edge.otherVertex == otherEdge.vertex) or
            (edge.otherVertex == otherEdge.OtherVertex))
            v2Check = TRUE
        endif
    endfor
    if      (v1Check = TRUE and v2Check == TRUE)
    continue
    endif
    else
        DeleteEdge(edge)
    endelse
endfor
```

5- Theorem (Euler): $v – e + f = 2$

A simple polygon has always $v=e$ and $f=2$ (interior and exterior).

Each vertex has degree greater than or equal to 3. Thus,

$3v <= 2e$ then $v <= 2/3 e$

$v – e + f – 2 = 0$ and $v <= 2/3 e$ then $f + 2/3 e – e <= f – 1/3 e$ then $e <= 3f – 6$

$v – e + f –2 = 0$ and $3/2 v <= e$ then $f + v – 3/2 v = f – 1/2 v$ then $v <= 2f – 4$

Each face touches at least 3 edges then $3 f <= 2e$ then $f <= 2/3e$

$v – e + f –2 = 0$ and $3/2 f <= e$ then $f + v – 3/2f = v – 1/2 f$ then $f <= 2v – 4$

$v – e + f – 2 = 0$ and $f <= 2/3 e$ then $2/3 e + v – e = v – 1/3 e$ then $e = 3v – 6$