Question 1. Minimum Spanning Trees  [12 marks]

Let $G = (V, E, w)$ be a undirected, connected, weighted graph and assume all weights are distinct (i.e., $w(e) \neq w(f)$ for any pair of edges $e, f \in E$ with $e \neq f$). Suppose $C_0$ is a given simple cycle in $G$, and $e_0 = (u_0, v_0)$ is an edge on $C_0$.

Given the above assumptions, for each of the questions below circle the most specific correct answer (i.e., if something “always” happens or “never” happens, then “sometimes” will be marked wrong).

Part (a)  [2 marks]
Suppose $e_0$ and $C_0$ are as above and $e_0$ is the maximum weight edge on $C_0$, then $e_0$ is

always sometimes never

in an MST of $G$.

Part (b)  [2 marks]
Suppose $e_0$ and $C_0$ are as above and the weight $w(e_0)$ is neither the minimum nor the maximum of the weights of the edges on $C_0$, then $e_0$ is

always sometimes never

in an MST of $G$.

Part (c)  [2 marks]
Suppose $e_0$ and $C_0$ are as above and $e_0$ is the minimum weight edge on $C_0$, then $e_0$ is

always sometimes never

in an MST of $G$.

Part (d)  [2 marks]
Suppose $e_1 \in E$ is the maximum weight edge over all of $E$. Then this $e_1$ (which may be different than the $e_0$ above) is

always sometimes never

in an MST of $G$.

Part (e)  [2 marks]
Suppose $e_0$ and $C_0$ are as above then it is

always sometimes never

ture that $|E| \geq |V|$.

Part (f)  [2 marks]
Consider an edge $a = (a_1, a_2) \in E$ such that there is no path $P = (p_1, p_2, \ldots, p_n)$ in $G$ such that $p_1 = a_1$, $p_n = a_2$, and all the edges on $P$ satisfy $w((p_i, p_{i+1})) < w(a)$ for $1 \leq i \leq n - 1$. Then $a$ is

always sometimes never

in an MST of $G$.

Total pages = 3
Question 2. Divide and Conquer: Count Duplicates [18 marks]

Given an unsorted list of integers, \( L \), we want to return both a sublist \( D \) which contains every integer in \( L \) but without any duplicates, and a list \( N \) of the same length as \( D \) for which \( N(k) \) equals the number of times the element \( D(k) \) appears in \( L \). For example, given \( L = (5, 3, 4, 3, 4, 3, 3) \), one possible output would be \( D = (3, 5, 4) \) and \( N = (4, 1, 2) \) (i.e., \( D \) can be given in any order, but the entries in \( N \) must correspond to \( D \)'s).

Part (a) [15 marks]

Finish the pseudo-code function \( \text{countDup} \) below for computing \( D \) and \( N \). To get any marks at all your algorithm must make essential use of the output from the recursive calls of \( \text{countDup} \), and must have a runtime of \( O(|L| \log |L|) \). Be precise, some marks will be taken off for incorrect details such as off-by-one errors. Include comments to assist the marker.

```plaintext
// For a list \( L \) of length \( n \), the solution \( D \) and \( N \) is given by
[D, N] = countDup(L, 1, n)

[D, N] = countDup(L, a, b)
// Input: Array \( L(1..n) \) of integers, and indices \( 1 \leq a \leq b \leq n \).
// Output: The lists \( D, N \) described above for the sublist \( L(k) \), \( k = a, \ldots, b \).
// Include any other important properties of the output lists \( D \) and \( N \).
//
// if a == b
   D = L(a); N = (1)
   return D, N
else
   m = floor((b + a - 1) / 2)
   [D1, N1] = countDup(L, a, m)
   [D2, N2] = countDup(L, m+1, b)
```

Total pages = 3
Page 2
CONT’D…
Part (b) [3 marks]
Analyze your algorithm’s running time.
(For your reference, the Master Theorem states that any function that satisfies a recurrence of the form
\[ T(n) = a T(n/b) + \Theta(n^d) \] has solution \( T(n) = \Theta(n^d) \) if \( a < b^d \), \( T(n) = \Theta(n^d \log n) \) if \( a = b^d \), and
\( T(n) = \Theta(n \log_b a) \) if \( a > b^d \).)