Justify your answers with proper reasonings/proofs.

1. Prove that any tree $T$ on $n$ vertices has a vertex $v$ such that if we remove $v$, then each connected component has at most $n/2$ vertices. Show how to find such a vertex in linear time.

2. You are given a tree $T$ where each vertex $v$ has an integer $val(v)$ stored in it (you can assume that all the integers involved are distinct). A vertex $v$ is said to be a local minimum if $val(v) \leq val(w)$ for all the neighbours $w$ of $v$. Show how to find a local minimum in $O(n)$ time, where $n$ is the number of vertices in $T$.

3. Solve the above problem when the graph is an $n \times n$ grid graph. An $n \times n$ grid graph has vertices labelled $(i, j)$, where $1 \leq i, j \leq n$ and $(i, j)$ is adjacent to $(i-1, j), (i+1, j), (i, j-1), (i, j+1)$ (with appropriate restrictions at the boundary). The time taken by the algorithm should be $O(n)$.

4. (a) Let $n = 2^l - 1$ for some positive integer $l$. Suppose someone claims to hold an unsorted array $A[1...n]$ of distinct $l$-bit strings; thus, exactly one $l$-bit string does not appear in $A$. Suppose further that the only way we can access $A$ is by calling the function $FB(i, j)$, which returns the $j$th bit of the string $A[i]$ in $O(1)$ time. Describe an algorithm to find the missing string in $A$ using only $O(n)$ calls to $FB$.

(b) Now suppose $n = 2^l - k$ for some positive integers $k$ and $l$, and again we are given an array $A[1...n]$ of distinct $l$-bit strings. Describe an algorithm to find the $k$ strings that are missing from $A$ using only $O(n \log k)$ calls to $FB$.

5. Suppose you are given an array $A[1...n]$ of numbers, which may be positive, negative, or zero, and which are not necessarily integers.

(a) Describe and analyze an $O(n)$ time algorithm that finds the largest sum of elements in a contiguous subarray $A[i...j]$.

(b) Describe and analyze an $O(n)$ time algorithm that finds the largest product of elements in a contiguous subarray $A[i...j]$.

6. Suppose you are given an array $M[1...n, 1...n]$ of numbers, which may be positive, negative, or zero, and which are not necessarily integers. Describe an algorithm to find the largest sum of elements in any rectangular subarray of the form $M[i...i', j...j']$ Your algorithm should run in $O(n^3)$ time.