Unless otherwise stated, assume that graphs are undirected, unweighted and simple (i.e., no self-loops and no multi-edges). As usual, \( n \) denotes the number of vertices and \( m \) the number of edges. **For all algorithms, explain time complexity and prove correctness.**

1. (20 pts) In a directed graph, a **get-stuck** vertex is one that has in-degree \( n - 1 \) and out-degree 0. Assuming that the adjacency matrix representation is used, design an \( O(n) \) algorithm to determine if a given graph has a get-stuck vertex. (Yes, this problem can be solved without even looking at the entire input matrix.)

2. (20 pts) Characterize all undirected connected graphs containing a vertex \( v \) such that there exists a DFS tree rooted at \( v \) that is identical to a BFS tree rooted at \( v \). (Two trees are identical if they contain the same set of edges; the order in which they are traversed is irrelevant. However, both trees must have the same root \( v \).) Prove your characterization is correct.

3. (20 pts) Given an unweighted tree \( T = (V, E) \), let \( \Gamma(T) \) denote the maximum distance between any two vertices in \( T \). Give an \( O(n) \) algorithm for computing \( \Gamma(T) \). For extra credit (5 additional pts), give an \( O(n) \) algorithm that works for unweighted and weighted trees.

4. (20 pts) Let directed graph \( G = (V, E) \) be a strongly connected graph and let \( T \) be a DFS tree in \( G \). Prove that if all the forward edges in \( G \), with respect to \( T \), are removed from \( G \), the resulting graph is still strongly connected.

5. **Ungraded.** A directed graph \( G = (V, E) \) is said to be **thinly connected** if \( u \leadsto v \) implies that there is at most one simple path from \( u \) to \( v \), for all \( u, v \in V \). Design an \( O(n^2) \) algorithm to test a graph for being thinly connected.