Problem 1: Consider the following Special Minimum Spanning Tree (SMST) problem.

**Given:** A connected graph $G(V, E)$ with a non-negative weight $w(e)$ for each edge $e \in E$ and a non-empty subset $L \subseteq V$.

**Requirement:** Determine whether $G$ has a spanning tree in which every node in $L$ appears as a leaf node. (The spanning tree may have other leaf nodes in addition to those in $L$.) If so, find such a spanning tree $T^*_L$ of minimum total weight.

As usual, let $|V| = n$ and $|E| = m$. Give an algorithm for the above problem that runs in time $O(m \log n)$ time. Your answer must include a high-level description of the algorithm, a proof of correctness and an analysis to establish the running time of the algorithm.

Problem 2: Let $G(V, E)$ be a connected undirected graph in which the weight of each edge is either 0 or 1. Explain how Prim’s algorithm can be implemented to find a minimum spanning tree of $G$ in $O(|V| + |E|)$ time.

Your answer should clearly specify the data structures used in your implementation, a modified outline of Prim’s algorithm that uses the data structures and an analysis to show that its running time is indeed $O(|V| + |E|)$.

Problem 3: Suppose $D(V, A)$ is a directed graph without self loops. Let $n$ denote $|V|$. Recall that for any vertex $v \in V$, the in-degree of $v$ is the number of edges entering $v$; the out-degree of $v$ is the number of edges leaving $v$. Let us call a node $v$ **special** if its in-degree is equal to $n - 1$ and its out-degree is zero.

Let $V = \{v_1, v_2, \ldots, v_n\}$. Suppose we use an $n \times n$ adjacency matrix $M$ to represent $D$. Here, $M[i, j] = 1$ if the directed edge $(v_i, v_j)$ is in $A$; and $M[i, j] = 0$ otherwise. Using this representation, give an $O(n)$ algorithm to determine whether $D$ has a special node. Your answer must include a clear description of the algorithm, its proof of correctness, and an explanation of why its running time is $O(n)$.

**Note:** Even though the adjacency matrix $M$ has $n^2$ entries, your algorithm can only look at $O(n)$ of those entries.