1. **Problem 4.9:** One of the basic motivations behind the Minimum Spanning Tree Problem is the goal of designing a spanning network for a set of nodes with minimum *total* cost. Here we explore another type of objective: designing a spanning network for which the *most expensive* edge is as cheap as possible.

   Specifically, let $G = (V, E)$ be a connected graph with n vertices, m edges, and positive edge costs that you may assume are all distinct. Let $T(V, E')$ be a spanning tree of G; we define the *bottleneck edge* of T to be the edge of T with the greatest cost.

   A spanning tree T of G is a *minimum-bottleneck spanning tree* if there is no spanning tree T' of G with a cheaper bottleneck edge.
   (a) Is every minimum-bottleneck tree of G a minumum spanning tree of G? Prove or give a counterexample.
   (b) Is every minimum-spanning tree of G a minimum-bottleneck tree of G? Prove or give a counterexample.

2. **Differences between Trees:** Suppose T_1 and T_2 are distinct MSTs for graph G. Let (u, v) be the lightest edge in T_1 but not in T_2, and (x, y) the lightest edge in T_2 but not T_1. Show that the weight of (u, v) is equal to the weight of (x, y).

3. **MST Uniqueness:** It is easy to see that the MST of a general graph need not be unique. Prove that if all the weights on the edges of a connected undirected graph are distinct, then there *is* a unique MST. You can use the claim of problem 2.

4. **Clustering:** Given a graph $G(V, E)$, a k-cluster is a partition of the vertices V into k disjoint sets, each of which is called a cluster. The distance between two clusters is defined as the smallest distance between any pair of vertices, one in each cluster. That is, for clusters U, W, $d(U, W) = min\{d(u, w) | u \in U, w \in W, (u, w) \in E\}$. Devise an algorithm that takes a graph $G(V, E)$ and an integer k, and partitions V into k clusters such that the minimum distance between any pair of clusters is maximized.