Problem 1. [20 points - No help question]
In your own words, expand on the following:

a) The worst-case running time complexity of Dijkstra’s algorithm. Briefly explain what data structures
would you use to implement the algorithm, and how they affect the running time complexity of the algorithm.
**Note:** You don’t need to rewrite the algorithm. Just mention the data structures and how they’d be used
within the algorithm.

b) The correctness of Prim’s algorithm. Explain how Prim’s algorithm can be used to find a minimum
spanning tree of a graph. Why does the algorithm successfully find a minimum spanning tree?

Problem 2. [15 points]
KT Chapter 4 - Problem 8. Suppose you are given an undirected connected graph G, with edge costs
that are all distinct. Prove that G has a unique minimum spanning tree.

Problem 3. [20 points]
KT Chapter 4 - Problem 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 minimum 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.

Problem 4. [25 points - Peer Review]
KT Chapter 4 - Problem 13. A small business, say a photocopying service with a single large machine,
faces the following scheduling problem. Each morning they get a set of jobs from customers. They want to
do the jobs on their single machine in an order that keeps their customers happiest. Customer $i$ job will
take $t_i$ time to complete. Given a schedule (i.e., an ordering of the jobs), let $f(i)$ denote the finishing time
of job $i$. For example, if job $j$ is the first to be done, we would have $f(j) = t_j$; and if job $j$ is done right after
job $i$, we would have $f(j) = f(i) + t_j$. Each customer $i$ also has a given weight $w_i$ that represents his or her
importance to the business.

The happiness of customer $i$ is expected to be dependent on the finishing time of $i$ s job. So the company
decides that they want to order the jobs to minimize the weighted sum of the completion times, $\sum_{i=1}^{n} w_i C_i$.

(a) Identify the input of the problem, and its goal.
(b) Design an efficient algorithm to solve this problem.
(c) Prove why it solves the problem efficiently.
Example. Suppose there are two jobs: the first takes time $t_1 = 1$ and has weight $w_1 = 10$, while the second job takes time $t_2 = 3$ and has weight $w_2 = 2$. Then doing job 1 first would yield a weighted completion time of $10 \times 1 + 2 \times 4 = 18$, while doing the second job first would yield the larger weighted completion time of $10 \times 4 + 2 \times 3 = 46$.

Problem 5. [20 points]

KT Chapter 4 - Problem 3. You are consulting for a trucking company that does a large amount of business shipping packages between New York and Boston. The volume is high enough that they have to send a number of trucks each day between the two locations. Trucks have a fixed limit $W$ on the maximum amount of weight they are allowed to carry. Boxes arrive at the New York station one by one, and each package $i$ has a weight $w_i$. The trucking station is quite small, so at most one truck can be at the station at any time. Company policy requires that boxes are shipped in the order they arrive; otherwise, a customer might get upset upon seeing a box that arrived after his make it to Boston faster. At the moment, the company is using a simple greedy algorithm for packing: they pack boxes in the order they arrive, and whenever the next box does not fit, they send the truck on its way.

But they wonder if they might be using too many trucks, and they want your opinion on whether the situation can be improved. Here is how they are thinking. Maybe one could decrease the number of trucks needed by sometimes sending off a truck that was less full, and in this way allow the next few trucks to be better packed.

Prove that, for a given set of boxes with specified weights, the greedy algorithm currently in use actually minimizes the number of trucks that are needed. Your proof should follow the type of analysis we used for the Interval Scheduling Problem: it should establish the optimality of this greedy packing algorithm by identifying a measure under which it stays ahead of all other solutions.