Problem 1.  [3 points - No help question]
a) The way that Dijkstra’s algorithm is written in page 138 of your textbook has a lot of details compressed into that first line of the body of the while loop. Rewrite the algorithm, making sure to clarify the exact operations performed within the body of the loop. You don’t need to go into the data structure details, or write in a Java-ish way. All you need to do is break down that down into smaller steps, and explain them in plain sentences.

b) The worst-case running time complexity of Dijkstra’s algorithm can be made to be Θ(mlogn), if we use the right data structures. Briefly explain what data structures would you use to implement the algorithm, and how they affect the running time complexity of the algorithm.

Problem 2.  [2 points]
For the following statement below, decide whether they are true or false. If a statement is true, give a short explanation. If it is false, give a counterexample.

Suppose we are given an instance of the Shortest s-t Path Problem on a directed graph G. We assume that all edge costs are positive and distinct. Let P be a minimum-cost s-t path for this instance. Now suppose we replace each edge cost $c_e$ by its square, $c_e^2$, thereby creating a new instance of the problem with the same graph but different costs. True or false? P must still be a minimum-cost s-t path for this new instance.

Problem 3.  [10 points - Proof Problem - Please submit proof separately in class]
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 $i$’s job will take $t_i$ time to complete. Given a schedule (i.e., an ordering of the jobs), let $C_i$ denote the finishing time of job $i$. For example, if job $j$ is the first to be done, we would have $C_j = t_j$; and if job $j$ is done right after job $i$, we would have $C_j = C_i + t_j$. 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) Design an efficient algorithm to solve this problem. That is, you are given a set of $n$ jobs with a processing time $t_i$ and a weight $w_i$ for each job. You want to order the jobs so as to minimize the weighted sum of the completion times, $\sum_{i=1}^n w_i C_i$.

(b) [Proof part] 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 $10 \times 1 + 2 \cdot 4 = 18$, while doing the second job first would yield the larger weighted completion time of $10 \times 4 + 2 \cdot 3 = 46$.

Problem 4.  [10 points]
Your friend is working as a camp counselor, and he is in charge of organizing activities for a set of junior-high-school-age campers. One of his plans is the following mini-triathlon exercise: each contestant must swim 20 laps of a pool, then bike 10 miles, then run 3 miles.
The plan is to send the contestants out in a staggered fashion, via the following rule: the contestants must use the pool one at a time. In other words, first one contestant swims the 20 laps, gets out, and starts biking. As soon as this first person is out of the pool, a second contestant begins swimming the 20 laps; as soon as he or she is out and starts biking, a third contestant begins swimming . . . and so on.

Each contestant has a projected swimming time (the expected time it will take him or her to complete the 20 laps), a projected biking time (the expected time it will take him or her to complete the 10 miles of bicycling), and a projected running time (the time it will take him or her to complete the 3 miles of running).

Your friend wants to decide on a schedule for the triathlon: an order in which to sequence the starts of the contestants. Let’s say that the completion time of a schedule is the earliest time at which all contestants will be finished with all three legs of the triathlon, assuming they each spend exactly their projected swimming, biking, and running times on the three parts.

Note that participants can bike and run simultaneously, but at most one person can be in the pool at any time.

(a) What are the inputs and goal of this problem? Try to formalize your notations, to help you later with the proof.
(b) What’s the best order for sending people out, if one wants the whole competition to be over as early as possible?
(c) Prove that your proposed order is optimal.

Problem 5. [7 points]
Remember your algorithm from last assignment to find a cycle in a graph with complexity of O(n+m). In this problem, give an algorithm that determines whether or not a given undirected graph \( G = (V, E) \) contains a cycle. Your algorithm should run in O(n) time only, independent of m.

(a) Provide pseudocode for your algorithm.
(b) Justify your algorithm’s running time.

Problem 6. [8 points]
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 7. [10 points]
Insertion sort can be expressed as a recursive procedure as follows. In order to sort \( A[1..n] \), we recursively sort \( A[1..n-1] \) and then insert \( A[n] \) into the sorted array \( A[1..n-1] \). Write pseudocode for this recursive version of Insertion sort, give a recurrence for the running time of the algorithm, and solve it using recurrence trees.