Problem 1.  [20 points - Programming question]

a) In your own words, expand on the following:

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) Implement Dijkstra’s algorithm using either Java or Python. Feel free to use any existing data structure library to help you with the implementation.

Problem 2.  [5 points]

For the following statements below, decide whether they are true or false. If a statement is true, give a short explanation. If it is false, give a counterexample.

a) 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.

b) True or false? If G is a directed graph that has a node with no incoming edges, then G is a DAG.

Problem 3.  [15 points - Proof Problem - Please submit separately]

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) **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 \( 10 \cdot 1 + 2 \cdot 4 = 18 \), while doing the second job first would yield the larger weighted completion time of \( 10 \cdot 4 + 2 \cdot 3 = 46 \).
Problem 4. [10 points]
For the upcoming bake sale, you are planning to bake $n$ different recipes, labeled $r_1, r_2, \ldots, r_n$. Recipe $r_i$ requires $p_i$ minutes of preparation time and $b_i$ minutes of baking time. Fortunately, you have access to a test kitchen with $n$ ovens, so all of recipes can bake simultaneously once they are prepared. However, there is only one of you, so you need to decide in which order to complete the preparation of each recipe.

For example: as soon as you complete preparing the first recipe, you can put it in the oven to bake and immediately begin preparing the second recipe. When you complete the second recipe, you can put it in the oven to bake whether or not the first recipe is done baking; and so on.

Let's say that a schedule is an ordering for preparation of the recipes, and the completion time of the schedule is the earliest time at which all recipes are done baking. This is an important quantity to minimize, because the bake sale is coming up soon!

Write the pseudocode of a polynomial-time algorithm that finds a schedule with as small a completion time as possible.