This assignment should be completed individually. You are allowed to discuss the problems together, but not discuss the answers. You must write the solutions on your own, and list the collaborators you worked with.

Submission instructions. This assignment is due by 9:00pm on Friday Nov 10, as a pdf file. Submit it to the PSet 6 folder in the CS231-assignments directory. Then, print it, and bring it to class the following Monday.

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 a 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. [20 points]

**KT Chapter 4 - Problem 13.** A small business, 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, $\sum_{i=1}^n w_i C_i$.

**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, $\sum_{i=1}^n w_i C_i$. **Briefly explain** why it solves the problem efficiently (Just explain, you don’t need to write a proof).

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 5. [30 points]

**KT Chapter 4 - Problem 18.** Your friends are planning an expedition to a small town deep in the Canadian north next winter break. They’ve researched all the travel options and have drawn up a directed graph whose nodes represent intermediate destinations and edges represent the roads between them.

In the course of this, they’ve also learned that extreme weather causes roads in this part of the world to become quite slow in the winter and may cause large travel delays. They’ve found an excellent travel Web site that can accurately predict how fast they’ll be able to travel along the roads; however, the speed of travel depends on the time of year. More precisely, the Web site answers queries of the following form: given an edge $e = (v, w)$ connecting two sites $v$ and $w$, and given a proposed starting time $t$ from location $v$, the site will return a value $f_e(t)$, the predicted arrival time at $w$. The Web site guarantees that $f_e(t) \geq t$ for all edges $e$ and all times $t$ (you can’t travel backward in time), and that $f_e(t)$ is a monotone increasing function of $t$ (that is, you do not arrive earlier by starting later). Other than that, the functions $f_e(t)$ may be arbitrary. For example, in areas where the travel time does not vary with the season, we would have $f_e(t) = t + l_e$, where $l_e$ is the time needed to travel from the beginning to the end of edge $e$.

Your friends want to use the Web site to determine the fastest way to travel through the directed graph from their starting point to their intended destination. (You should assume that they start at time 0, and that all predictions made by the Web site are completely correct.) **Give a polynomial-time algorithm** to do this, where we treat a single query to the Web site (based on a specific edge $e$ and a time $t$) as taking a single computational step. **Show why it’s correct.**

Problem 6. [20 points]

**KT Chapter 5 - Problem 1.** You are interested in analyzing some hard-to-obtain data from two separate databases. Each database contains $n$ numerical values so there are $2n$ values total and you may assume that no two values are the same. Youd like to determine the median of this set of $2n$ values, which we will define here to be the $n$th smallest value. However, the only way you can access these values is through queries to the databases. In a single query, you can specify a value $k$ to one of the two databases, and the chosen database will return the $k$th smallest value that it contains. Since queries are expensive, you would like to compute the median using as few queries as possible.

**Give an algorithm** that finds the median value using at most $O(\log n)$ queries.
Problem 7. [15 points]
CLRS Exercise 2.3-4. 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, and give a recurrence for the running time of the algorithm, and solve it.

Problem 8. [15 points]
Solve the following recurrences.

Assume $T(n) = 1$ for $n < 2$. Express your solutions in Big-Oh notation. Show your work. (Note that you can use any of the methods discussed in class.)

a) $T(n) = 4T(n/2) + n$

b) $T(n) = T(n/2) + 1$

c) $T(n) = T(n - k) + 1$, $k \geq 0$