• Write proofs clearly and concisely – points will be deducted otherwise.

• Pseudo-code is not required for algorithms – just a clear description with all the key ideas and proof of correctness will suffice.

• For NP-Completeness proofs, here are some NP-complete problems you may choose to reduce from: SATISFIABILITY, SET-COVER, KNAP-SACK, HAMILTONIAN-PATH; you may assume that these problems are in fact NP-complete.
Problem 1. [25 points] Test taking strategy:

a. **Case of no partial credit:** Consider the following scenario. You are taking an exam with \( n \) problems. After quickly scanning the exam, for every question \( i = 1, \ldots, n \), you know that it will take you \( t_i \) time to solve it, and it will give you \( q_i \) points if you solve it, with no credit for any partially solved problems. The total amount of time you have to take the exam is \( T \). You want to maximize your number of points. Can you solve the problem in polynomial time, or is it NP-hard? If the former, give an efficient algorithm for solving it. If the latter, prove that it is NP-hard.

b. **Case of allowed partial credit:** Now consider the same scenario, but we allow credit for each partially solved problem. Specifically, if you spend \( t'_i \) (with \( 0 \leq t'_i \leq t_i \)) time on question \( i \), you will get \( q_i t'_i / t_i \) points. Can you solve the problem in polynomial time, or is it NP-hard? If the former, give an efficient algorithm for solving it. If the latter, prove that it is NP-hard.
Problem 2. [25 points] James is planning to drive to city X from home. The trip will be following a highway, with James’s home at coordinate 0 and city X at coordinate $T$ (you can assume both James’s home and city X are directly adjacent to the highway). James’s car can drive a total of 400 miles with a full tank of gas, and he leaves his home with a full tank. There are $n$ gas stations along the way, the exit to $i$-th gas station is at coordinate $x_i$ on the highway (where we have $0 < x_1 < x_2 < \cdots < x_n < T$), and to get to the $i$-th gas station, there is a detour of $t_i$ miles (that means it takes $t_i$ miles to get from the exit to the gas station, and also $t_i$ miles to get back to the same exit). James wants to plan his trip so that the total lengths of the detour he takes is minimized (going to gas station $i$ counts as $2t_i$ in the length of the detour). James always fills up his tank at every gas station. You are going to design a dynamic programming algorithm to solve this problem. You can think of the destination as the $n + 1$-th gas station with $x_{n+1} = T$ and $t_{n+1} = 0$; similarly you can think of James’s home as 0-th gas station with $x_0 = 0$ and $t_0 = 0$.

a. Design an $O(n^2)$ algorithm for solving the problem.

b. Design an $O(n)$ algorithm for solving the problem. **Hints**: Use dynamic programming. Prove that if a later gas station becomes more preferable than an earlier gas station, it will stay that way, so one can keep a queue of gas stations that are still in competition. Prove that every station will enter and exit the queue at most once. Apply amortized analysis to show the total running time of this improved algorithm is $O(n)$. 
Problem 3. [25 points] Internet company X has been using a network of wired connections to connect its customers. The network consists of \( n + 1 \) vertices (0, 1, 2, ..., \( n \)) and \( m \) edges. Vertex 0 represents the internet access point of company X and vertices 1, 2, ..., \( n \) are the customers. Each edge \((u, v)\) can be activated for a cost of \( w_{u,v} \). A customer has internet connection as long as it has a path to vertex 0 using only activated edges. Note that the input to the problem only specifies the edge weights, but not which edges are activated.

Now company X is considering replacing some of the links with wireless connections. There are \( k \) different wireless technologies. Technology \( i \) has a base cost of \( c_i \) and a per-connection cost of \( l_i \). That is, if company X pays the base cost of \( c_i \), then it can establish a wireless connection from vertex 0 to any customer for the cost of \( l_i \) (even if there were no original wired connection between 0 and that customer). With wireless connections, a customer has internet connection, if it has a path to vertex 0 or any vertex that has established a wireless connection (it is also OK if the vertex itself has established a wireless connection). The technologies are listed in the order where \( c_1 > c_2 > \cdots > c_k \), and then \( l_1 < l_2 < \cdots < l_k \).

Your task is to help company X decide which wireless technology (if any) would be the most cost efficient. The total cost includes the base cost \( (c_i) \), wireless connection costs \( (l_i) \) per wireless connection) and the activation cost for all the wired connections. More precisely, design an algorithm that outputs the lowest total cost. Note that the company can only choose to not use any wireless technology or use one of the technologies (using multiple technologies does not really decrease the cost). There is always a way to serve all customers even without using an wireless technology.

As an example, given input Technology Graph of Figure 1 with 7 wired connections that can be activated among 4 customers and the company, each with a cost specified on the edge. There are \( k = 2 \) technologies: tech 1 with \( c_1 = 10 \) and \( l_1 = 1 \), and tech 2 with \( c_2 = 2 \) and \( l_2 = 3 \). A solid line represents an activated wired connection. A dotted line represents an established wireless connection. In this case your algorithm should output 13.
Problem 4. [25 points] You are a gate agent responsible for seating passengers in an airplane. All the $n$ passengers are lined up in front of you and there is an airplane that has $n$ seats that you need to board them on. Every passenger has a pre-assigned seat (that you do not know until the passenger reaches the front of the line and is boarded), except the first passenger in line (let us call him Bob) whose ticket was confirmed at the last minute and he wasn’t assigned a seat number. Your first task is to assign a seat for Bob. Subsequently, when any passenger with a pre-assigned seat boards the airplane (let us call any such passenger Alice), she unseats Bob from her seat if Bob is occupying her seat, and you must then find a new empty seat for Bob. Your goal is to design an algorithm to seat (and re-seat if required) Bob so that the number of times he changes his seat is minimized.

a. Show that any deterministic algorithm can end up assigning every seat on the airplane to Bob, i.e., change his seat $n - 1$ times.

b. Design and analyze a randomized algorithm that only changes Bob’s seat $O(\log n)$ times in expectation.