1. (40 points) You wish to drive from point A to point B along a highway minimizing the time that you are stopped for gas. You are told beforehand the capacity C of your gas tank in liters, your rate $F$ of fuel consumption in liters/kilometer, the rate $r$ in liters/minute at which you can fill your tank at a gas station, and the locations $A = x_1, \ldots, B = x_n$ of the gas stations along the highway. So if you stop to fill your tank from 2 liters to 8 liters, you would have to stop for $6/r$ minutes. Consider the following two algorithms:

**Algorithm A:** Stop at every gas station, and fill the tank with just enough gas to make it to the next gas station.

**Algorithm B:** Stop if and only if you don’t have enough gas to make it to the next gas station, and if you stop, fill the tank up all the way.

For each algorithm either prove or disprove that this algorithm correctly solves the problem. Your proof of correctness must use an exchange argument. I want to know whether you know how an exchange argument works, so for full credit you must give the full exchange argument, as well as explaining what the exchange is and why it is correct.

2. (20 points) We consider the problem of computing the longest increasing subsequence of a sequence $x_1, \ldots, x_n$ of distinct integers. We considered this problem several times during class.

(a) Let $LIS(k)$ be the longest increasing subsequence of $x_1, \ldots, x_k$. Prove that $LIS(k)$ can not be computed from $LIS(k-1)$ and $x_k$. Be very clear in explaining how you prove such a claim. What I want to know is whether you understand how to show that naive recursion won’t work.

(b) We showed in class that you can recursively compute $LIS(k, j)$ where $LIS(k, j)$ is the length of the longest increasing subsequence of $x_1, \ldots, x_k$ with some property $P(j)$. What was this property $P(j)$?

(c) Give a recursive formula for computing $LIS(k, j)$.

(d) Consider the algorithm that you obtain by converting this recursive algorithm into an iterative, array-based, bottom-up algorithm. Show the table/array that this iterative algorithm constructs on the input

$$9, 10, 15, 1, 6, 4, 8$$

3. (20 points) The input to this problem is a pair of strings $A = a_1 \ldots a_m$ and $B = b_1 \ldots b_n$. The goal is to convert $A$ into $B$ as cheaply as possible. The rules are as follows. For a cost of 3 you can delete any letter. For a cost of 4 you can insert a letter in any position. For a cost of 5 you can replace any letter by any other letter. For example, you can convert $A = ababca$ to $B = abacab$ via the following sequence: $ababca$ at a cost of 5 can be converted to $abaabc$, which at cost of 3 can be converted to $ababce$, which at cost of 3 can be converted to $abace$, which at cost of 4 can be converted to $abacac$, which at cost of 4 can be converted to $abacab$. Thus
the total cost for this conversion would be 19. This is almost surely not the cheapest possible conversion. Give an $O(n^2)$ time dynamic programming algorithm for finding the cheapest way to convert the strings. Your algorithm doesn’t need to find the actual operations.

You must include some English description as to why your algorithm is correct, and a definition of the meaning of the entries in the array that you use.