1  Maximum consecutive subarray

You are given a sequence $S$ of integers (not necessarily positive). The number of integers is $n$. The problem is to find the consecutive subsequence of $S$ with maximum sum. Here, “Consecutive” means that you are not allowed to skip numbers. For example if the input was 12, −14, 1, 23, −6, 22, −34, 13 the output would be 1, 23, −6, 22.

Now give a linear time dynamic programming algorithm for this problem.

You should explain why a naive recursive solution is not possible. That is, figure out why knowing the $i$-th number, and the maximum consecutive sum of the first $i - 1$ numbers, is not enough to compute the maximum consecutive sum of the first $i$ numbers. This should give you a big hint how to strengthen the inductive hypothesis.

2  Comparing two sequences

This is the common task for spell check. You are given two sequences, $S_1$ and $S_2$, with $n$ and $m$ symbols each. You want to find out how “similar” these two sequences are. One way is to add spaces into $S_1$ and $S_2$ so that the two sequences become the same length. For example, let $S_1$ be ANNDREW and $S_2$ be AMDREWS. We can add one space to each:

ANNDREW-
A-MDREWS

Here, - stands for the added space. Intuitively, adding spaces makes the two sequences look similar and then we can compare them. We then score the two changed sequences as follows: we compare the corresponding positions (i.e. $S_1[i]$ with $S_2[i]$); if the two symbols match, the cost is 0; if the two mismatch (possibly one of them is -), the cost is 1. Note, both symbols are -, the cost is 0. In the above example, the total cost of this comparison is equal to 3. Our goal is to find the smallest cost over all feasible ways of adding spaces into $S_1$ and $S_2$.

Now, give a dynamic programming algorithm for this problem. Write the pseudo-code for your algorithm, then explain why it works. Also give an analysis of running time. This problem is related to the LCS problem. So try to define subproblems similar to those of the LCS problem.

3  Interleaving strings

Do problem 19 of Chapter six (page 329). Briefly, in this problem, you are given three strings $s$, $x$ and $y$. The goal is deciding whether $s$ is an interleaving of $x$ and $y$. We say $s$ is an interleaving of $x$ and $y$ if symbols in $s$ can be partitioned (not necessarily contiguously) into two subsequences $s'$ and $s''$ such that $s'$ is repetition of $x$ and $s''$ is repetition of $y$.

Now give a polynomial time algorithm for this problem, which takes strings $s$, $x$ and $y$ and decides if $s$ is an interleaving of $x$ and $y$. 
