This document is to provide how you would answer a question from lecture if it were homework or exam questions instead. Notes in red would not appear in the homework submission, but are for commentary. The text below, in black, would be sufficient for full credit if the LCS problem, presented in lecture, were instead a homework/exam question.

Give a clear English description of what you are trying to evaluate; do not refer to the internal state ("the current indices") or *how* the algorithm makes this computation (that will be described later). In general, I would prefer that you give the function a meaningful name in the context of the problem rather than naming it something like "OPT." Your function should not be a single variable name either, although sometimes I provide input parameters that are.

Define LCS(i, j) to be the longest common subsequence between $X[1 \dots i]$ and $Y[1 \dots j]$. This could alternately be phrased as "…the first i characters of X and the first j characters of Y."

Give a *correct* recursive algorithm. This can either be explicit, such as via a recursive algorithm (shown here) or with iterative pseudo-code. If you provide the latter, you do not need to give the recursive solution separately, and the memoization method and evaluation order should be clear from your pseudo-code.

```
LCS(i, j)
  if i == 0 or j == 0 then
    return 0
  if X[i] == Y[j] then
    return 1+ LCS(i − 1, j − 1)
  return max(LCS(i, j − 1), LCS(i − 1, j))
```

Unless stated otherwise, when we ask you for a dynamic programming algorithm in this class, it is sufficient to compute the value or cost of an optimal solution. For example, in LCS, you do not need to provide the common subsequence itself, merely the length of the longest one. A good reinforcement exercise for you would be to write the code that produces it from the resulting table, but I am not collecting that.

Lastly, give the details of the iterative algorithm: how do you memoize, in which order do you fill in the vector, and how long does your iterative algorithm take? The first two might be implicit if you write your solution iteratively, although in that case, you must explicitly point out the recursive computation.

This can be stored in a two-dimensional vector, LCS$[0 \dots n, 0 \dots m]$, where n and m are the lengths of the two input strings X and Y, respectively. This can then be filled in first by the base cases, and then the general cases can be filled in by increasing value of the first parameter, then increasing value of the second. Each recursive case takes $\mathcal{O}(1)$ to fill in, and there are $\mathcal{O}(nm)$ cases, for a total of $\mathcal{O}(mn)$ time.