Layla Oesper, Jeff Ondich

One question of your choice (either #1 or #2 or #3) is due by **5:00PM Tuesday, May 14** and will be graded on completion only (1 point). Your full answers to all questions are due by **5:00PM Friday, May 17**. Each solution will be graded for correctness and clarity (4 points per question). Read the course information page/syllabus for further info about this 4-point scale.

0. Estimate the amount of time you spent on each question and include it at the top of your solution. Also, list your collaborators for each question at the top of your solution.

1. The *Northfield Gazette* is a worldwide viral sensation for its wacky take on local news (don't miss the video of cows grazing set to *Call Me Maybe*, which overwhelmed YouTube's servers during the summer 2013). For the past 15 years, the *Gazette*'s web analytics team (i.e., a guy named Derek) has collected daily counts of unique visitors to its website. The marketing team just realized they have access to this historical data, and they're now talking excitedly about "actionable business insights" and "drilling down into the bleeding edge of best practice blue-sky thinking".

   At the marketers' request, Derek has created a list A of daily changes measured in unique visitors. For example, a portion of A looks like $[..., 35000, -12000, 14000, 3000, -2000, ...]$, which says that one day the visitor count went up by 35000, the next day back down by 12000, then up by 14000 the next day, and so on. The marketing team wants Derek to find the contiguous sequence of days over which the best net change occurred. They could then match that stretch of days to the news stories and PR initiatives issued by the *Gazette* to see if they can figure out how to replicate those past successes.

   You are Derek's algorithms-savvy friend, and he has asked for your help. He would like you to **give a divide-and-conquer algorithm** to determine the maximum value of the sum of a contiguous sequence of entries in A. For example, if $A = [-2, -5, \mathbf{6, -2, -3, 1, 5}, -6]$, your algorithm should return 7, corresponding to the boldfaced sequence of daily visitor-count changes.

   Provide your algorithm, proof of correctness, and runtime analysis.

   Note: You may be able to think of non-divide and conquer algorithms for this, but for full credit you must submit a solution that is divide and conquer.

2. Dynamic programming is good at counting all sorts of things. Consider, for example, a rectangular grid m squares wide and n squares high, using a coordinate system that puts $(1, 1)$ in the top left and (m, n) in the bottom right. You have a coin sitting in the upper-left square, and your goal is to move it to the bottom-right square by some combination of legal moves. The only legal moves are "move one square to the right" and "move one square down".

   This is obviously not a challenging game to play, but here's a more interesting question: given m and n, how many different sequences of legal moves are there to get from the top left to the bottom right? Let's answer this question using dynamic programming.

   Let $M(w, h)$ represent the number of legal move sequences there are to get the coin from square $(1, 1)$ to square (w, h). Using this notation:

   (a) Write a recurrence relation, including base case(s), expressing $M(w, h)$ in terms of smaller subproblems of the same type.

   (b) Describe an algorithm for computing $M(m, n)$ in small-degree polynomial time. You do not need to provide a correctness proof for this problem, though of course your algorithm needs to be correct.

(c) Analyze the runtime of your algorithm.

3. A lot of jazz and pop music is written down in "chart" form–that is, just the melody plus a chord sequence. Based on these elements, the performer has some structure, but a lot of freedom to choose exactly which notes to play and when.

   It turns out that many songs share similar chord sequences. Jazz songs in C major, for example, often contain a sequence of chords $[C^6, A^{-7}, D^{-7}, G^7, C^6]$. (By the way, even though music theory is cool, you definitely do not need to know what these symbols mean to solve this problem!)

   Suppose you have a database consisting of the full sequence of chords in each of a large number of songs, and you want to learn about the common chord sequences that appear in these songs to help you practice. One operation you might want to build into your song-analysis software is this: given the full chord sequence for each of two songs, find the longest shared contiguous subsequence of chords.

   To simplify matters, let's imagine we have given each possible chord a one-letter name. Starting with the chords listed above, we might say $A = C^6$, $B = A^{-7}$, etc., so that the sequence above would be represented by $[A, B, C, D, A]$. Further, let's just try to find the **length** of the longest shared contiguous subsequence rather than the subsequence itself. For example, if your two songs are $[A, B, C, D, A, D, A]$ and $[E, F, G, F, C, D, A, F, D, A, G]$, the length of the longest shared contiguous subsequence is 3, for $[C, D, A]$.

   Write a dynamic programming algorithm to compute, given two sequences of chords (one of length m and the other of length n), the length of the longest shared contiguous subsequence. As usual, provide an argument for your algorithm's correctness and analyze its runtime.