CMSC 423 Homework #2:

Due: Oct. 6th at the start of class

You may discuss these problems with your classmates, but you **must write up your solutions independently**, without using common notes or worksheets. You must indicate at the top of your homework who you worked with. Your write up should be clear, concise, and neat. You are trying to convince a skeptical reader that your algorithms or answers are correct. Messy or hard-to-read homeworks will not be graded.

1. A *monotonically increasing subsequence* of a sequence of n integers $c_1, \ldots, c_n$ is a subset of the integers $c_{i_1}, \ldots, c_{i_k}$ such that $c_{i_1} < c_{i_2} < \cdots < c_{i_k}$ and $i_1 < i_2 < \cdots < i_k$. In other words, it is a subsequence of the input that is always increasing when considered in the order present in the input. For example: $50, 75, 100$ is the longest increasing subsequence of input $80, 50, 75, 35, 100$.

   Give an $O(n^2)$ dynamic programming algorithm to find the *longest* monotonically increasing subsequence of a given set $c_1, \ldots, c_n$ of integers.

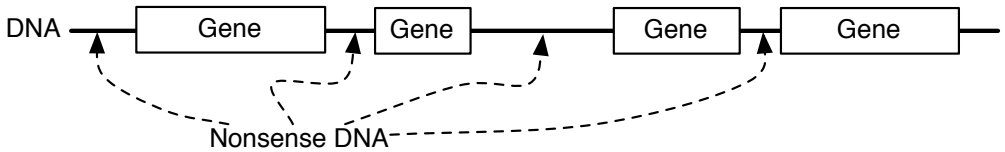
2. You are given strings x and y, both of length n, and a positive integer k. We are interested in the best global alignment between x and y using a simple gap penalty. Notice that because $|x| = |y|$ the gaps must come in pairs, one in x and one in y.

   Suppose the user guarantees that an optimal global alignment between x and y will have no more than k gap pairs. Sketch an $O(kn + n)$ dynamic programming algorithm to compute this optimal global alignment. For this problem, you don't need to provide a full algorithm, just the main idea.

   (Hint: consider first when $k = 0$.)

3. Suppose you are given strings x and y, match, mismatch, and gap costs m, s, and g, and an integer $M \geq 0$ that is your mismatch budget. Give an efficient dynamic programming algorithm that finds the best global alignment between x and y **that uses no more than M mismatches** (it can use as many gaps or *matches* as it needs to). Explain the running time of your algorithm.

4. You are given a very long string S of {A,C,G,T}s representing the genome of the bed bug (an actual organism researchers here at UMD have studied — they assembled its genome to find genes that could be targeted for better pest control). You know that some subsequences of various lengths of this genome encode for genes, and these gene sequences are separated by some amount of junk, nonsense DNA. Here's a picture:



   However, we don't know where each gene starts or ends. We are just given the long string S. Researchers have developed a function called **GeneScore**(s) that we can give any string s and that returns a high number if the s is likely a gene and returns a low (maybe negative) number if s is not likely a gene. We want to partition the string S into some unknown number k of non-overlapping genes $g_1, \ldots, g_k$ such that $\sum_{i=1}^{k}$ **GeneScore**(g_i) is as large as possible.

   Give a dynamic programming algorithm to compute the choice of boundaries of the genes that maximizes the sum of the **GeneScore**s for the identified genes.