Due at 11:59PM, Friday, April 25th on Canvas. Submissions must be ***typed*** and in PDF format. Show your work for full credit!

1. Run the dynamic programming algorithm, including ***reconstruction***, to find a ***longest common subsequence*** of the DNA sequences/strings

   TCGGCGTAGA and CAAGCATATGG.

   Show the table that is generated by the algorithm, and the sequence of steps needed to reconstruct an LCS. Your final answer should be ***string***, and not just its length. (Note: this question is ***not*** asking for any code whatsoever)

2. In class, I've made the following claim: given a set of coin denominations, if each denomination is a multiple of the previous one, then the cashier's algorithm is always optimal. In this problem you will prove this claim for currencies with three denominations. Consider a currency with coins of value $1, a, ab$, where a and b are integers bigger than 1. (We saw this in class for the U.S. currency where $a = 5$ and $b = 2$)

   (a) Show that, in any optimal solution, the number of 1-cent coins is at most $(a - 1)$.

   (b) Show that, in any optimal solution, the number of a-cent coins is at most $(b - 1)$.

   (c) Use parts (a) and (b) to prove that, for any amount of change $C \geq ab$, every optimal solution for C has an ab-cent coin. (Hint: as we did in class, prove the contrapositive)

3. For which of the following currencies is the cashier's algorithm ***always optimal***? If it's always optimal, give a brief justification, and if it's not always optimal, give a counterexample.

   (a) 1, 13, 60

   (b) 1, 14, 98

   (c) 1, 9, 63, 315

   (d) 1, 7, 19, 45

4. Construct Huffman codes for the following sets of characters and their frequencies. Draw the Huffman tree (you may attach photos of your trees), write down the codewords for each character, and calculate the total length of the encoded string:

   (a) A 17, B 25, C 36, D 31, E 4, F 9, G 33, H 16

   (b) S 8, T 38, U 2, V 1, W 6, X 4, Y 36, Z 14

   For the sake of having a unique solution, in each iteration of Huffman's algorithm, make the left child the "heavier" of the two trees, i.e., the one with larger frequency.