Instructions:
Include pseudo-code for your algorithms, a run-time analysis and a proof of correctness.

Also, read these instructions carefully: [http://courses.cs.washington.edu/courses/cse417/16wi/grading.html](http://courses.cs.washington.edu/courses/cse417/16wi/grading.html)

Turn your homework in on canvas: Problems 1-3 in one file, and problems 4-6 in the other.

1. Huffman coding:
   - Simulate the Huffman tree algorithm on the input alphabet a, b, c, d, e with frequencies .15, .30, .50, .02, .03, resp. Show the intermediate "trees" built by the algorithm, as well as the final tree.
   - What will be the total length in bits of the compressed text built from a sequence of 100 letters having the above frequencies using the Huffman code you calculated?

2. Problem 6.1 on page 181 of [DPV]. Be sure your algorithm outputs the optimal sequence.

3. Problem 6.4 on page 182 of [DPV].

4. Problem 6.8 on page 183 of [DPV]. Be sure your algorithm outputs the positions \((i, j)\) where the longest common substring starts, and the length \(k\) of this longest common substring.

5. Design and implement a dynamic programming that computes a mistype distance between words \(w_1\) and \(w_2\) that reflects the chance of typing word \(w_2\) when word \(w_1\) is intended. Your distance function will need to take into the account errors including mistyping characters, adding characters, deleting characters, and transposing characters. The errors of mistyping should also reflect the distance between characters on a key board, as it is more likely to type an s for an a than it is to type a p for an a. (You don’t need to include all possibilities for distance between characters on a keyboard. Just include a few, say, for the 5 most common letters.) You will need to use your judgment in defining the parameters for this algorithm. For this problem describe your distance function and how it computes the mistype distance. Include the documented code for your dynamic programming algorithm.

6. Run your algorithm on the inputs provided (which will be available by the evening of Wednesday, March 2) and turn in your results. The input will consist of lists of words (one per line), where you are to compute the distance from the first word in the list to all of the other words. One particular input (group of words) will be separated from the next by a line containing ****
Here is an example of what the input might look like.

https://courses.cs.washington.edu/courses/cse421/15au/homework/hw8_input.txt

Your output should be the input augmented with the computed distances for each word.