APS Homework 4: Dynamic Programming
Optional Challenge Problems

Problem 1: Longest Common Subsequence

Given two strings *s* and *t*, a *common subsequence* of *s* and *t* is a list of characters that appear in the same order in both *s* and *t* (but not necessarily contiguously). For example, if *s* = “FAST” and *t* = “FURIOUS”, the sequence [‘F’, ‘S’] is a common subsequence of *s* and *t* because both characters appear in this order in both strings (“**F**A**S**T” and “**F**URIOU**S**”). The empty list, [], is a valid subsequence, so if two strings have no characters in common (e.g. *s* = “VIN” and *t* = “DOM”), their only common subsequence is the empty list, [].

Problem 1a: Given two arbitrary strings *s* and *t*, describe a Dynamic Programming algorithm for computing the Longest Common Subsequence (LCS) of *s* and *t*.

***Problem 1b*:** Prove that the algorithm you provided in *Problem 1a* is correct for any arbitrary two strings *s* and *t*.

Problem 2: Edit Distance

Imagine I have three possible “edits” I can perform on strings: “insertion” (adding a character), “deletion” (removing a character), and “substitution” (replacing a character with something else). I can “transform” a string *s* into a string *t* by performing a sequence of edits. For example, to transform *s* = “KITTEN” into *t* = “SITTING”, I can do the following:

“**K**ITTEN” → “**S**ITTEN” (substitution)
“SITT**E**N” → “SITT**I**N” (substitution)
“SITTIN” → “SITTIN**G**” (insertion)

Given two strings *s* and *t*, let the *Edit Distance* of *s* and *t* be the smallest number of edits required to transform *s* into *t*.

Problem 2a: Describe a Dynamic Programming algorithm to compute the Edit Distance of two strings *s* and *t*.

***Problem 2b*:** Prove that the algorithm you provided in *Problem 2a* is correct for any arbitrary strings *s* and *t*.