Dynamic Programming

CSCI 112, Lab 6

File names: Names of files, functions, and variables, when specified, must be EXACTLY as specified. This includes simple mistakes such as capitalization.

Individual work: All work must be your own. Do not share code with anyone other than the instructor and teaching assistants. This includes looking over shoulders at screens with the code open. You may discuss ideas, algorithms, approaches, *etc.* with other students but NEVER actual code. Do not use code written by anyone else, in the class or from the internet.

Documentation: Each file should begin with a docstring that includes your name, the class number and name, the lab number, and a short description of the lab, as well as documentation pertinent to that particular file.

Substring: One string is a *substring* of another if the characters in the first string occur, in order, somewhere in the second string. For example, the string "geoff" is a substring of the string "xxgxxeoxxfxxxf". We will write a boolean function that returns `True` or `False` depending on whether the first argument is a substring of the second.

Recursive solution: Code up a recursive solution to this problem using the following strategy:

- If the first string is empty then return `True`
- If the second string is empty then return `False`
- If the first letter of the first string does **not** match the first letter of the second string, then recursively try to match the first string with the second string without its first letter.
- If the first letter of the first string **does** match the first letter of the second string, then check recursion both with and without matching the first letters. If either one succeeds, then the case succeeds.

Dynamic programming: Code up a non-recursive solution to this problem by building up an $m \times n$ table of results matching each letter of the potential substring with the superstring. For example, trying to determine whether "abc" is a substring of "xabbxc" we get the following table, where 1 is `True` and 0 is `False`:

		c	a	b	b	x	c
	1	1	1	1	1	1	1
a	0	0	1	1	1	1	1
b	0	0	0	1	1	1	1
c	0	0	0	0	0	0	1

Each cell in the table represents whether the string at the left (up to the current row) is a substring of the string at the top (up to the current column). This can be determined from the two current characters in the strings, and the entries in the table above and to the left of the current entry, so the table can be filled in with a dynamic programming algorithm.

Turn in: Two procedures: `is_substring_recursive` and `is_substring_dynamic`, defined in a file called `substring.py`. Also a unittest module called `substring_test.py` with randomly generated test strings. Put all in a folder called `csci112lab06yourname`, zip and turn into canvas.