Lab 16: Recursion

Part 0: Clone the Repo

Again, since the numbering is a bit weird, you'll find the repo at
`git clone https://github.com/physics91si/username-recursion.git recursion`

Part 1: Fibonacci with cache

Remember the Fibonacci sequence from Lab 2? Of course you do: `fib(n)=fib(n-1)+fib(n-2)`, and `fib(0)=0` and `fib(1)=1`. Now we'll implement it more efficiently, using a cache.

To compare, you can write out our old-fashioned fibonacci function real fast. You'll see that a number as low as, say, `fib(40)` will make your computer have to think quite a bit. But since we know most of the "thinking" is just recalculating the same results over and over again, we can speed it up dramatically if we store intermediate results in a cache. **Your task is to write an efficient fibonacci function with a cache.** You can implement the cache as a global dictionary.

If you play around with this function, you'll note that it works quite fast, until your inputs give you
`RuntimeError: maximum recursion depth exceeded in comparison`
Python actually lets you change the maximum recursion depth. For example,

```
import sys
sys.setrecursionlimit(5000000)
```

But you should be careful, because for every level of recursive call, it takes up a little more space in memory. Once you get deep enough, you'll start getting segfaults, and in this context, it usually means that the program should be rewritten with a while-loop.

Part 2: Levenshtein Distance

The Levenshtein distance is a metric for determining how different two words are, in terms of the minimum number of changes we need to take one word from the other, where a change is defined as a single character insertion, deletion, or substitution. For example,

- `lev("armchair", "armhair") == 1` because we make one change: delete **c**.
- `lev("physics", "psychics") == 3` because we make three changes: substitute **h**->**s**, substitute **s**->**c**, then insert **h**.

Your task is to write the recursive function lev().

Hint: You can express the Levenshtein distance between two strings in terms of the Levenshtein distance between the strings with the last letter chopped off of one or both of them.