A Recursion Checklist I

Now that you've seen several examples of recursion, let's apply recursion to a few problems, similar to those on a programming exam. To help you, here is a **checklist** that will help you identify the most common sources of errors**.**

☐ Check the Simple (Base) Cases

Does your recursive implementation begin by checking for simple base-cases? First check to see if the problem is so simple that no recursion is necessary. Recursive functions always start with the keyword `if`; if yours doesn't, look carefully.

☐ Have You Solved the Base Cases Correctly?

A surprising number of bugs in arise from **incorrect solutions to the simple base-cases**. If the simple cases are wrong, the rest of the function will inherit the same mistake. If you had defined `factorial(0)` as returning `0` instead of `1`, **any** argument would end up returning `0`.

☐ Does Decomposition Make it Simpler?

Does your recursive decomposition **make the problem simpler**? Problems must get simpler as you go along; the problem **must get smaller** as it proceeds. If the problem does not get simpler, you have the recursive analogue of the infinite loop, which is called **nonterminating or infinite recursion**.