Recursion

Have you ever seen a set of Russian dolls? At first, you see just one figurine, usually painted wood, that looks something like this:

You can remove the top half of the first doll, and what do you see inside? Another, slightly smaller, Russian doll!

You can remove that doll and separate its top and bottom halves. And you see yet another, even smaller, doll:

And once more:

And you can keep going. Eventually you find the teeniest Russian doll. It is just one piece, and so it does not open:

We started with one big Russian doll, and we saw smaller and smaller Russian dolls, until we saw one that was so small that it could not contain another.

What do Russian dolls have to do with algorithms? Just as one Russian doll has within it a smaller Russian doll, which has an even smaller Russian doll within it, all the way down to a tiny Russian doll that is too small to contain another, we'll see how to design an algorithm to solve a problem by solving a smaller instance of the same problem, unless the problem is so small that we can just solve it directly. We call this technique **recursion**.

Recursion has many, many applications. In this module, we'll see how to use recursion to compute the factorial function, to determine whether a word is a palindrome, to compute powers of a number, to draw a type of fractal, and to solve the ancient Towers of Hanoi problem. Later modules will use recursion to solve other problems, including sorting.