Wrappers & Helpers

We can instantly calculate `fib(n)` when we know the values of `fib(n-1)` and **`fib(n-2)`**. When you don’t know those values, the calculation takes a lot of time, but when you do, then it’s really fast.

Are there values for `fib(n-1)` and `fib(n-2)` that we **do know**? Let’s write out the sequence and see:

n	0	1	2	3	4	5	6	7	8	9	10	...
fib(n)	0	1	1	2	3	5	8	13	21	34	55	...

Since `fib(0)` is `0` and `fib(1)` is `1`, we can start there. For our **recursive helper**, just write a function that accepts `n` and the two terms: `t0` and `t1`. If `n` is `0` return `t0` and if it is `1`, return `t1`. Otherwise call the function recursively with `n - 1`.

When calling the function recursively, however, instead of passing the `t0` and`t1`, calculate the **next two terms** and pass those instead. Here’s what the helper should look like:

```
int helper(int n, int t0, int t1)
{
    if (n == 0) return t0;
    if (n == 1) return t1;
    return helper(n - 1, t1, t0 + t1);
}
```

For the `fib()` **wrapper function**, just call the helper, kick-starting it with the first two terms, `0` and `1`. Here’s what the function looks like:

```
int fib(int n)
{
    return helper(n, 0, 1);
}
```