Recursive factorial

For positive values of **n**, let's write **n!** as we did before, as a product of numbers starting from n and going down to **1: n!** = **n·(n−1)⋯2·1**. But notice that **(n−1)⋯2·1** is another way of writing **(n−1)!**, and so we can say that **n!** = **n·(n−1)!**. Did you see what we just did? We wrote **n!** as a product in which one of the factors is **(n−1)!**. We said that you can compute **n!** by computing **(n−1)!** and then multiplying the result of computing **(n−1)!** by **n**. You can compute the factorial function on n by first computing the factorial function on **n−1**. We say that computing **(n-1)!** is a subproblem that we solve to compute **n!**.

Let's look at an example: computing **5!**.

- You can compute **5!** as **5·4!**.
- Now you need to solve the subproblem of computing **4!**, which you can compute as **4·3!**.
- Now you need to solve the subproblem of computing **3!**, which is **3·2!**.
- Now **2!**, which is **2·1!**.
- Now you need to compute **1!**. You could say that **1!** equals **1**, because it's the product of all the integers from **1** through **1**. Or you can apply the formula that **1!** = **1·0!**. Let's do it by applying the formula.
- We defined **0!** to equal **1**.
- Now you can compute **1!** = **1·0!=1**.
- Having computed **1!** = **1**, you can compute **2!** = **2·1!** = **2**.
- Having computed **2!** = **2**, you can compute **3!** = **3·2!** = **6**.
- Having computed **3!** = **6**, you can compute **4!** = **4·3!** = **24**.
- Finally, having computed **4!** = **24**, you can finish up by computing **5!** = **5·4!** = **120**.

So now we have another way of thinking about how to compute the value of n!, for all nonnegative integers n:

- If **n** = **0**, then declare that **n!** = **1**.
- Otherwise, **n** must be positive. Solve the subproblem of computing **(n−1)!**, multiply this result by **n**, and declare **n!** equal to the result of this product.

When we're computing **n!** in this way, we call the first case, where we immediately know the answer, the **base case**, and we call the second case, where we have to compute the same function but on a different value, the **recursive case**.