Solution Review: Tail Recursion

In the following lesson, we will go over the solution of the challenge: Tail Recursion.

We'll cover the following

Task

In this challenge, you had to create a tail-recursive factorial function.

Solution

A skeleton of the function was already provided for you. Let's look it over.

```
def factorial(x: Int): Int = {
  def loop(accumulator: Int, x: Int): Int = {


  }
  loop(1,x)
}
```

As before, `factorial` takes a single parameter `x` of type `Int`. However, this time, its function body consists of a nested function `loop`. `loop` is the tail recursive part of `factorial`. It has two parameters `accumulator` and `x`. The accumulator stores the current value of the factorial in each recursive call. This is why when we pass **1** as the initial accumulator when `loop` is called in the function body of factorial.

```
loop(1,x)
```

1 is the smallest possible factorial.

Hence, if the number whose factorial we want to find is **0**, we will simply return the accumulator as is, this is our base case.

```
if(x == 0) accumulator
```

The recursive case is if `x` is not equal to **0**. We will recursively call loop in this case. The new `accumulator` to be passed will be `accumulator * x` and the new `x` to be passed will be `x-1`.

```
else loop(accumulator*x,x-1)
```

As the last thing being done by `loop` is a recursive call, it is a tail-recursive function.

You can find the complete solution below:

> You were required to write the code on **line 3** and **line 4**.

This code requires the following environment variables to execute:

LANG	C.UTF-8

```
def factorial(x: Int): Int = {
  def loop(accumulator: Int, x: Int): Int = {
    if(x==0) accumulator
    else loop(accumulator*x,x-1)
  }
  loop(1,x)
}

// Driver Code
print(factorial(5))
```

Let’s wrap up this chapter with a quiz to test what you have learned so far.