Challenge: Tail Recursion

Test yourself and implement what you have learned so far in this challenge.

We'll cover the following

- Problem Statement
  - Input
  - Output
  - Sample Input
  - Sample Output
  - Test Yourself

Problem Statement

In a previous lesson on recursion, we implemented the recursive factorial function. You need to design a tail-recursive version of the factorial function given below.

This code requires the following environment variables to execute:

LANG	C.UTF-8

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

// Driver Code
print(factorial(3))
```

> The tail recursive version of factorial requires a nested function `loop`. `loop` has two parameters: `accumulator` and `x` and is the tail recursive part of factorial. All you have to do is figure out what the function body of `loop`

should contain.

Input

The input of the function is a number `x` of type `Int` whose factorial value you want to compute.

Output

The output will be the factorial of `x`.

Sample Input

```
5
```

Sample Output

```
120
```

Test Yourself

Write your code in the given area. Try the exercise by yourself first, but if you get stuck, the solution has been provided. Good luck!

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 = {
    // Write your code here

    return -1 // Remove this line after writing your code
  }
  loop(1,x)
}
```

Hint 1 of 2

The `accumulator` keeps track of the current factorial in each

recursive call.

Let’s go over the solution review in the next lesson.