

ASSIGNMENT 2 – HIGHER-ORDER FUNCTIONS

Advanced programming paradigms

In this assignment, you will work with tail recursive functions and higher-order functions.

Question 1 – Tail recursion

- (a) Define a tail-recursive version of `fact`
- (b) The Fibonacci function is defined as follows:

$$fib(x) = \begin{cases} 0, & \text{if } x = 0 \\ 1, & \text{if } x = 1 \\ fib(x-1) + fib(x-2), & \text{otherwise} \end{cases}$$

You are asked to write two implementation of this function.

- 1) The first one using the definition above.
- 2) A second solution using tail recursion. For this one, do not forget to check that your function is tail-recursive with the `@tailrec` annotation¹.

Question 2 – Higher-order functions

- (a) The sum function we defined during the course and that computes

$$\sum_{i=a}^b f(x)$$

uses a linear recursion. Can you transform it to a tail recursion by filling the ??? hereafter?

```

1  def sum(f: Int => Int, a: Int, b: Int) = {
2      def iter(a: Int, acc: Int): Int = {
3          if (???) ???
4          else iter(???, ???)
5      }
6      iter(???, ???)
7  }
```

Question 3 – Currying

- (a) Using the sum function as a source of inspiration, write a function `product` that computes the product of the values of a function for the integers in a given interval, i.e.

$$\prod_{i=a}^b f(i)$$

Make sure that this function is in its curried form.

- (b) Write `factorial` in terms of the function product that you defined in part a.
- (c) Write a more general function that generalizes both sum and product. This done, provide a new implementation of `sum`, resp. `product`, using that new function.

¹For which you have to import `scala.annotation.tailrec`