## TUTORIAL 3 (SOLUTIONS)

1. This Scala function calculates the factorial of an integer using recursion:

def **factorial**(n: Int): Int = {  
 if (n <= 1)  
 n  
 else   
 **factorial**(n - 1) \* n   
}

It be called like this to find the factorial of 4:

factorial(4)

You can trace the way the function works by tracing the recursive calls and their return values using a table like the one below. The function calls are made recursively until the stopping condition is reached, then the last call returns its value to the next-to-last, and so on.

|  |  |  |
| --- | --- | --- |
| **Function call** | **Stopping condition (n<=1)** | **Returned value** |
| factorial(4) | false | 3\*4 = 24 |
| factorial(3) | false | 2\*3 = 6 |
| factorial(2) | false | 1\*2 = 2 |
| factorial(1) | true | 1 |

Similarly, this Scala function reverses a string using recursion:

def reverse(str:String):String = {  
 if (str == null || str.equals(""))  
 str  
 else  
 reverse(str.substring(1)) + str.substring(0, 1)  
}

reverse(“hello”)

gives

“olleh”

Trace the way this function works by writing out a table similar to the factorial example

|  |  |  |
| --- | --- | --- |
| **Called** | **Stopping condition** | **Returned value** |
| reverse(“hello”) | false | “hello” |
| reverse(“ello”) | false | “olle” |
| reverse(“llo”) | false | “oll” |
| reverse(“lo”) | false | “ol” |
| reverse(“o”) | false | “o” |
| reverse(“”) | true | “” |

1. This is a tail-recursive version of the factorial function.

def factorial (n: Int): Int = {

def **fact**(n: Int, accumulator: Int): Int = {  
 if (n <= 1)  
 accumulator  
 else  
 **fact**(n - 1, n\* accumulator)  
 }

fact(n 1)  
}

* 1. What is the advantage of using tail-recursive functions?
  2. How can you tell from the code that the function is tail-recursive?
  3. Trace the way this function works by writing out a table similar to the first factorial example. Add an extra column in the table to trace the value of the variable *accumulator* within each call. Deduce the purpose of *accumulator*.

|  |  |  |  |
| --- | --- | --- | --- |
| **Called** | **Stopping condition** | **accumulator** | **Returned value** |
| factorial(4) |  |  | 24 |
| fact(4,1) | false | 1 | 24 |
| fact(3,4) | false | 4 | 24 |
| fact(2,12) | false | 12 | 24 |
| fact(1,24) | true | 24 | 24 |

1. How would you modify the reverse function to make it tail-recursive?

def reverse(str:String):String = {  
 def **rev**(str:String, accumulator:String):String = {  
 if (str == null || str.equals(""))  
 accumulator  
 else  
 **rev**(str.substring(1), str.substring(0, 1) + accumulator)  
 }  
  
 rev(str, "")  
}

|  |  |  |  |
| --- | --- | --- | --- |
| **Called** | **Stopping condition** | **accumulator** | **Returned value** |
| reverse(“hello”) | false |  | “olleh” |
| rev(“hello”,””) | false | “” | “olleh” |
| rev(“ello”,”h”) | false | “h” | “olleh” |
| rev(“llo”,”eh”) | false | “eh” | “olleh” |
| rev(“lo”,”leh”) | false | “leh” | “olleh” |
| rev(“o”,”lleh”) | false | “lleh” | “olleh” |
| rev(“”,”olleh”) | true | “olleh” | “olleh” |