

School of Humanities and Informatics

## WRITTEN EXAMINATION

Course      Advanced Programming

Sub-course

Course code IT732A

Credits for written examination    5,5

Date          20191213

Examination time 3h

Examination responsible    Gunnar Mathiason

Teachers concerned        Elio Ventocilla

Aid at the exam/appendices

Other

Instructions

<input type="checkbox"/>
<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>

Take a new sheet of paper for each teacher.

Take a new sheet of paper when starting a new question.

Write only on one side of the paper.

Write your name and personal ID No. on all pages you hand in.

Use page numbering.

Don't use a red pen.

Mark answered questions with a cross on the cover sheet.

Grade points

**Examination results should be made public within 18 working days**

*Good luck!*

# Advanced Programming - IT732A HT19

## Exam

University of Skövde

December 13, 2019

### Rules

- All questions are to be answered within the context of functional programming.
- You are expected to answer in a thorough, yet concise manner. That is, elaborate on your answers without dwelling on aspects which are not strongly related to the question at hand.
- Code examples are to be written in Scala code. Syntax mistakes will be overlooked.
- Write in an intelligible manner. If the hand writing needs to be decoded, no points will be awarded.
- **The exam is strictly individual.**
- The exam is composed of 5 questions, each with a value of 20 pts., adding to a total of 100 pts. A minimum of 50 pts. is required to pass.

#### Question 1.

- a) Which are the principles of functional programming? (8 points).
- b) Define them and give concrete examples for each of the principles (12 points).

#### Question 2.

Write a function `count`, which counts the times a digit appears within a given number.

- a) Write it in an **non**-tail-recursive form (6 points).
- b) Write it in a **tail**-recursive form (7 points).
- c) Write it as a **val**-defined function in a **curried** form (7 points).

Note: assume that all digits are  $> 0$ . Hint: recall that the modulo operator (%) returns the remainder of a division, e.g.,  $12\%10 = 2$ ;  $553\%10 = 3$ .

```
count(1, 1433115)  // = 3
count(4, 1433115)  // = 1
val countOnes = count(1)    // (curried)
countOnes(1433115)  // = 1
```

#### Question 3.

Given the following two functions:

```
def foo(n: Int, ns: List[Int], acc: Int = 0): (Int, List[Int]) = {
  if (ns == Nil) (acc, Nil)
  else if (ns.head != n) (acc, ns)
  else foo(n, ns.tail, acc + 1)
}
```

```
def las(ns: List[Int]): List[Int] = {
  if (ns == Nil) Nil
  else {
    val t = foo(ns.head, ns)
    t._1 :: ns.head :: las(t._2)
  }
}
```

- a) Decompose the call `las(List(1,2,1,1))` using the substitution model and give the result (10 points).
- b) Rewrite both functions using pattern matching (10 points).

#### Question 4.

Maps are data structures which associate elements of one type (keys) to elements of the same or another type (values). Table 4 shows an example of key-value pairs, where keys are of type String and values are of type Int.

key	value
"one"	1
"december"	12
"false"	0
"true"	1

Concretely, a Map is a collection of key-value pairs, where keys are always unique. Recall functional data structures and abstract data types (ADT), and do the following:

- a) Define a **functional** data structure Map which represents the collection previously described. Your definition should work for String keys and Int values (6 points).
- b) Define a method `add` as a part of your Map structure, which takes a key (String) and a value (Int), and returns a new Map with the key-pair added. Note that if the key exists, the value should be "replaced" (6 points).
- c) Define functional data structures and ADTs. **Relate to your solution.** (8 points).

#### Question 5.

The Hofstadter  $G$  sequence is given by  $G(0) = 0$ , and then  $G(n) = n - G(G(n-1))$  when  $n > 0$ . These are the first 15 elements in the sequence:

0, 1, 1, 2, 3, 3, 4, 4, 5, 6, 6, 7, 8, 8, ..

- a) Write a function `hofstadter` which takes a number  $n$ , and returns the Hofstadter element at the  $n$  position (8 points).
- b) Write a function `hofstadterStream` which returns a Stream of all numbers in the Hofstadter  $G$  sequence. (12 points).

Example calls:

```
hofstadter(0)    // = 0
hofstadter(2)    // = 1
hofstadter(8)    // = 5
hofstadterStream().take(5).toList // = List(0, 1, 1, 2, 3)
```