UNIVERSITY
OF SKÖVDE

School of Humanities and Informatics

WRITTEN EXAMINATION

Course Advanced Programming

Sub-course

Course code IT732A

Credits for written examination 5.5 hp

Date 20181108

Examination time 4 hours

Examination responsible Elio Ventocilla

Teachers concerned

Aid at the exam/appendices

Other

Instructions

- ☐ 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 F (0 - 49), E (50 - 54), D (55 - 61), C (62 - 82), B (83 - 94), A (95 - 100)

Examination results should be made public within 18 working days

Good luck!

Total number of pages

Advanced Programming - IT732A HT18
Exam

University of Skövde

November 8, 2018

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. Small 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.
Critically reflect about differences between functional programming and imperative programming. What elements sets them apart? What possible advantages and disadvantages might there be between one and the other?

Question 2.
What is referential transparency? How is it related to the substitution model?
Additionally, given the following function:

```
def foo(n: Int): List[Int] = {
  if (n <= 1) Nil
  else {
    val n1 = if (n % 2 == 0) n / 2 else (3 * n) + 1
    n1 :: foo(n1)
  }
}
```

Decompose the call `foo(5)` by hand using the substitution model and show the result.

Question 3.
Explain first-order and high-order functions. Write a code example that illustrates of both first-class and high order functions. Give comments to the code.

Question 4.
What is currification? From your point of view, when could it be useful?

Additionally, write a currified version of the following function using function literals (i.e. `val` defined):

```
val getY = (i: Int, m: Int, x: Int) => m * x + i
```

Question 5.

Finding the root (or zero) of a function means finding a value x such that $f(x) = 0$. The Newton method is a method for finding the root of a given function. It is based on the following recurrence relation:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

where f is the function whose root we want to find, and f' is the derivative of this function.

Write a function `root` that returns a stream (infinite structure) with all the elements of this sequence. This function will take an initial point x, a function f and its derivative f'. The signature of this function should be:

```
def root(x: Double, f: Double => Double, d: Double => Double): Stream[Double]
```

so that the following call can be made

```
val f = (x: Double) => x * x - 8.0
val d = (x: Double) => 2 * x
root(10, f, d)  // = Stream[Double]
```