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	20181219	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

December 19, 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 i.e. to motivate your answer. 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 tail-recursion? What benefit does it have with respect to normal recursion? Convert the following recursive function to its tail-recursive version:

```
def foo(n: Int): Int = {
  if (n <= 1) 1
  else n * foo(n - 1)
}
```

Question 3.
What makes a functional data structure, functional? Define a functional data structure for stacks with two functions: `push` and `pop`. Function `push` adds an element to the top of stack while `pop` takes, and removes, the top element. Remember that it should be a *functional* data structure. In-built functions, and other Scala structures such as lists, are not allowed.

Example of the expected use of the stack:

```
val a = new Stack(3)      // [3]
val b = a.push(4)         // [4, 3]
val (x1, c) = b.pop()     // (4, [3])
val (x2, d) = c.pop()     // (3, [])
```

Question 4.
Explain the properties of *monoids*, *monads* and *functors*, and how these might relate to effective parallel operations.

Question 5.
The following function signature provides a general way to create streams:

```
def unfold[A, B](s: B, f: B => Option[(A, B)]): Stream[A]  = ???
```

The `s` parameter represents an initial state. The `f` parameter, on the other hand, is a function that takes a state and returns an `Option` element with a tuple of two values: an element of the stream, and the following state with which the next element of the stream is to be computed. If the `f` function produces `None`, then the stream terminates.

Write the body of the function `unfold` and then make a call to it such that a stream of natural numbers (1, 2, 3, ...) is produced.