Durham University

Examination Paper

Examination Session:	Year:	Exam Code:
Model Exam	XXXX	COMP2221-WE01

Title: Programming Paradigms – Submodule Functional Programming

Time Allowed:	1 hour	
Additional Material provided:	None	
Materials Permitted:	None	
Calculators Permitted:	No	
Visiting Students may use dictionaries:	Yes	

Instructions to Candidates:	Answer ALL questions. Please answer each section in a separate answer booklet.

Section A Functional Programming (Laura Morgenstern)

Except where otherwise stated, any code you write in this section must be in Haskell.

Question 1

(a) Haskell uses **lazy evaluation**. Describe how this differs from **eager evaluation** (as seen in C# or C) with reference to functions and their arguments. **[4 Marks]**

(b) Consider an operation `scan` which computes the prefix sum on lists of arbitrarily large integers. When given a list $[x_0, x_1, x_2, \ldots, x_{n-1}]$, `scan` should return the list $[y_0, y_1, y_2, \ldots, y_{n-1}]$ where $y_0 = x_0$, $y_1 = x_0 + x_1$, and generally $y_j = \sum_{i=0}^{j} x_j$. Implement `scan` recursively and make use of pattern matching in your answer. **[10 Marks]**

(c) Now turn `scan` into a polymorphic, higher-order function. Rewrite `scan` as a new function, `scanf`, which accepts an additional argument that can be any binary operator. Ensure that `scanf` continues to yield the results of `scan` if you pass in `(+)` as the higher-order argument. **[4 Marks]**

Question 2

(a) Provide type annotations for the following functions. Include type constraints where required.

- A function `func1` that takes a string as input and returns the string reversed
- A function `func2` that takes two arbitrarily large integers and returns both as a pair
- A function `func3` that takes a list of arbitrary numerical parameters and returns their sum

[8 Marks]

(b) Consider the following data type `Letter` that represents either a lowercase letter `Minuscule` or an uppercase letter `Majuscule`.

```
data Letter = Minuscule Char | Majuscule Char
```

Write a function `isLowercase` which returns `True` if the letter is a `Minuscule` and `False` otherwise. **[3 Marks]**

(c) Explain the concept of functors in Haskell and write a `Functor` instance for the `Maybe` data type:

```
data Maybe a = Just a | Nothing
```

[6 Marks]

Question 3

(a) Reduce the λ-expression $(\lambda x.(xx))((\lambda y.(ay))b)$ to normal form. **[4 Marks]**

(b) In the λ-calculus, functions take exactly one argument. In practice, however, we often require higher-order functions with multiple parameters. Write down a λ-expression that can model a higher-order function with two input parameters. How is this technique called? **[4 Marks]**

END OF PAPER