CSC4020Z: Functional Programming

Practical Assignment 1: Haskell Basics

Department of Computer Science
University of Cape Town, South Africa

Due: Friday, 8th March, 2019, 11.55 PM

Assignment Instructions and Description

The *Glasgow Haskell Compiler* (GHC) provides an interactive interpreter (GHCi), which will be the main Haskell tool used in this module. The usual way to write Haskell programs is to have two windows open: one for a text editor to write your code, and the other for GHCi so that you can regularly load and test your code. For example, a Haskell script defining the following function:

```
double x = x + x
```

And named: *script1.hs* can be compiled via typing:

```
ghci script1.hs
```

GHCi should load and you should see something like:

```
...
[1 of 1] Compiling Main          ( script1.hs, interpreted )
Ok, one module loaded.
*Main>
```

In this case *script1* can then be tested via typing the function name and some value, for example:

```
double 7
```

Given this, implement Haskell functions that provide solutions to the following **four (4)** computational problems. Submit your scripts in a single ZIP file via Vula, using your student number as the ZIP file name (e.g.: XYZZYX001.ZIP) and each script named according to the corresponding question (e.g.: *question1.hs*, *question2.hs*, ...).

1. Define a function product that produces the product of a list of numbers. For example: *product [2,3,4]*, should produce the solution: *24*. **(20%)**
2. The library function *last* selects the last element of a non-empty list; for example: *last [1,2,3,4,5] = 5*. Write another definition for the *last* function in terms of the other library functions. **(20%)**
3. Using library functions, define a function *halve :: [a] − > ([a],[a])* that splits an even length list into two halves. For example: *halve [1,2,3,4,5,6]*, should produce: *([1,2,3],[4,5,6])*. **(20%)**
4. Consider a function *safetail :: [a] − > [a]* that behaves in the same way as the *tail* function except that it maps the empty list to itself rather than producing an error. Using *tail* and the function *null :: [a] − > Bool* that decides if a list is empty or not, define *safetail* using only: (1) a conditional expression, (2) guarded equations, and (3) pattern matching. Defining three different functions for solving these three different problems is an acceptable approach. **(40%)**

Note: Values in bold parentheses are the percentage weighting of each question as a portion of the total assignment mark.