CSC4020Z: Functional Programming

Practical Assignment 2: Haskell

2021

Department of Computer Science
University of Cape Town, South Africa

DUE: Friday, 23rd of April, 2021, 5.00 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

Implement *Haskell* functions that provide solutions to the following computational problems given in each of the two (2) parts of this assignment:

Part A: Five (5) questions: 10 marks.

Part B: One (1) questions: 10 marks.

Submit your scripts in a single ZIP file via *VULA* assignments tab, using your student number as the ZIP file name (e.g.: XYZZYX001.ZIP) and each script named according to the corresponding part and question number (e.g.: *partA-question1.hs*, *partA-question2.hs*, ... , *partB-question1.hs*).

Part A [10 Marks]

1. Consider these two *Haskell* functions (*bf*, *df*) which modify every other element of their list argument:

   bf,df :: [Int] → [Int]

   bf [ ] = [ ]

   bf [x] = [abs x]

   bf (x:y:xs) = (abs x) : y : bf xs

   df [ ] = [ ]

   df [x] = [x + 1]

   df (x:y:xs) = (x + 1) : y : df xs

   Define a more general function *gf :: (a → a) → [a] → [a]* such that the following property holds:

   ```
   propgf xs = bf xs == gf abs xs
               && df xs == gf (+1) xs
   ```

   (2 marks)

2. The following data-types are used to represent a hand of cards:

```
data Suit = Hearts | Clubs | Diamonds | Spades
      deriving Eq
data Rank = Numeric Int | Jack | Queen | King | Ace
      deriving Eq
data Card = NormalCard Rank Suit | Joker
      deriving Eq
```

Define a function:

$$countAces :: [Card] \rightarrow Int$$

Where: *countAces* returns the number of cards in the given hand which are either aces or jokers. For example, if there are 3 aces and 2 jokers in the hand, the answer will be 5.

(4 marks)

3. Define a function $sort :: [Int] -> [Int]$ that sorts a list of integers into numeric order, using a sorting method of your choice.

(2 marks)

4. Define a function $cp :: [[a]] -> [[a]]$ that returns the *Cartesian product* of a list of lists.

   For example, $cp\ [[1,2,3],[4,5,6]]$ should return.:

   $[[1,4],[1,5],[1,6],[2,4],[2,5],[2,6],[3,4],[3,5],[3,6]]$

   (1 mark)

5. Define a recursive function $nat2int :: Nat -> Int$ that converts a natural number to the corresponding integer.

   (1 mark)

Part B [10 Marks]

1. Design and implement a *Turing machine* that operates only on the following input and output symbols: $\{a,\ b\}$ and computes a function $f(x)$, where $f(x)$ outputs only the symbol a if x is a palindrome and outputs the symbol b otherwise. A palindrome is a symmetrical string, i.e., if we reverse the order of its symbols, it is still the same string.

   For example:

   $f(bba) = b$

   $f(baab) = a$

   $f(abab) = b$

   $f(aaa) = a$

   $f(babab) = a$

   The input string x can be up to length $N = 4$, but must contain only the symbols: a, b. The Turing machine terminates after the last symbol in the string of symbols has been processed or if it is given an empty string { }. In the latter case it terminates immediately with the output a.

   In a text-file (named README), give the sequence of possible Turing machine state configurations that are used during the computation of f with the input string: aba

   The list of possible configurations should be in the following format:
   $< CurrentState, InputSymbol, OutputSymbol, NewState >$

* * *