CSC4020Z: Functional Programming

Practical Assignment 2: Haskell

Department of Computer Science
University of Cape Town, South Africa

Due: Monday, 25th March, 2019, 11.55 PM

Assignment Description

Implement Haskell functions that provide solutions to the following **six (6)** 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. Suppose that arithmetic expressions built up from integers, addition and multiplication are represented using the following types:

   data Expr $=$ *Val Int* | *App Op Expr Expr*
   data Op $=$ *Add* | *Mul*

   Define functions:

   eval :: *Expr* $\rightarrow$ *Int*
   values :: *Expr* $\rightarrow$ *[Int]*

   that respectively evaluate an expression to its integer value, and return the list of integer values contained in an expression.

   (12%)

2. Define a function:

   delete :: *Int* → *[Int]* → *[Int]*

   that deletes the first occurrence (if any) of a value from a list. For example, *delete* 2 [1, 2, 3, 2] should give the result [1, 3, 2].

   (12%)

3. Using *delete*, define a function:

   perms :: *[Int]* → *[[Int]]*

   that returns all permutations of a list, given by all possible re-orderings of its elements. For example, *perms* [1, 2, 3] should return:

   [[1, 2, 3], [1, 3, 2], [2, 1, 3], [2, 3, 1], [3, 1, 2], [3, 2, 1]]

   (20%)

4. Define a function:

   split :: *[Int]* → *[([Int], [Int])]*

   that returns all splits of a list into two non-empty parts that append to give the original list. For example, *split* [1, 2, 3, 4] should return:

   [([1], [2, 3, 4]), ([1, 2], [3, 4]), ([1, 2, 3], [4])]

   (16%)

5. Using *split*, define a function:

   exprs :: *[Int]* → *[Expr]*

   that returns all expressions whose list of values is a given list. For example, *exprs* [1, 2, 3] should return all e for which $values\ e = [1, 2, 3]$.

   (24%)

6. Using your answers to the previous parts, define a function:

   solve :: *[Int]* $\rightarrow$ *Int* $\rightarrow$ *[Expr]*

   that returns all expressions whose list of values is a permutation of the given list and whose value is the given value.

   For example, *solve* [1, 2, 3, 4] 10 should return all expressions e for which *values* e is a permutation of [1, 2, 3, 4] and *eval* $e = 10$.

   (16%)

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