National Undergraduate Programme in Mathematical Sciences
National Graduate Programme in Computer Science

Functional Programming in Haskell

End-semester Examination, I Semester, 2019–2020

Date : November 29, 2019 Marks : 100
Time : 0930 – 1230 Weightage : 40%

---

1. Define the function `inits :: String -> [String]` that returns all the *initial segments* of the input string, in order of increasing length.

   Sample cases:

   ```
   inits ""          = [""]
   inits "abcde"     = ["", "a", "ab", "abc", "abcd", "abcde"]
   ```

   make O(n)

   (10 marks)

2. Define the function `tails :: String -> [String]` that returns all the *end segments* of the input string, in order of decreasing length.

   Sample cases:

   ```
   tails ""          = [""]
   tails "abcde"     = ["abcde", "bcde", "cde", "de", "e", ""]
   ```

   (

   (10 marks)

3. Define a function `firstRep :: String -> Maybe Char` that returns the first character that repeats in a string. If all characters are distinct, then it should return `Nothing`.

   Sample cases:

   ```
   firstRep ""            = Nothing
   firstRep "abcde"       = Nothing
   firstRep "aabcdde"     = Just 'a'
   firstRep "abcdebadc"   = Just 'b'
   firstRep "abbbbaba"    = Just 'b'
   ```

   (15 marks)

4. Recall the binary search tree data type presented in class. Recall also that in a binary search tree, all nodes in the left subtree have value less than that of the root, and all nodes in the right subtree have value greater than that of the root.

```
data BST a = Nil | Node (BST a) a (BST a)
```

Define a function `path :: Ord a => a -> BST a -> [a]` that takes a value and a tree as input, and finds the path from root to value in the tree (if the value is in the tree). If the value is not in the tree, the function returns [].

Sample cases:

```
path 5 Nil                                             = []
path 5 (Node (Node Nil 2 Nil) 3 (Node Nil 5 Nil))  = [3,5]
path 5 (Node (Node Nil 2 Nil) 3 (Node Nil 4 Nil))  = []
path 4 (Node (Node Nil 2 Nil) 3
       (Node (Node Nil 4 (Node Nil 5 Nil)) 6 Nil)) = [3,6,4]
```

(15 marks)

5. This question is about representation of matrices. Assume the following data definitions:

```
type Val = [[Int]]
data Matrix = Matrix {numRows :: Int, numCols :: Int, entries :: Val}
```

`entries` is a list of rows, each row being a list of `Int`s, `numRows` is the number of rows, and `numCols` is the number of columns.

Write a program `isValid :: Matrix -> Bool` that verifies whether an input matrix is valid, i.e., the number of rows and columns are positive, and the `entries` list actually has the number of rows and columns advertised.

(10 marks)

6. We can represent a sequence of matrix multiplications by a tree, as explained below.

Assume the `BinTree` definition given below (which represents binary trees with values attached only to the leaves).

```
data BinTree a = Leaf a | Fork (BinTree a) (BinTree a)
```

We can use a `BinTree Matrix` to represent a particular sequence of multiplying matrices. For instance, the matrix multiplication `((m1*m2)*(m3*m4))*m5` can be represented by the following tree.

```
Fork (Fork (Fork (Leaf m1) (Leaf m2)) (Fork (Leaf m3) (Leaf m4))) (Leaf m5)
```

Write a function `multipliable :: BinTree Matrix -> Bool` which checks whether the input tree represents a valid multiplication. In the above example, the sequence of multiplications is possible if the number of columns in `m1` is the same as the number of rows in `m2`, ..., and the number of columns in `m4` is the same as the number of rows in `m5`.

Sample cases:

```
m1 = Matrix {numRows = 2, numCols = 5, entries = ll1}
m2 = Matrix {numRows = 5, numCols = 3, entries = ll2}
m3 = Matrix {numRows = 3, numCols = 2, entries = ll3}

multipliable (Leaf m3)                                      = True
multipliable (Fork (Fork (Leaf m1) (Leaf m2)) (Leaf m3)) = True
multipliable (Fork (Leaf m1) (Fork (Leaf m2) (Leaf m3))) = True
multipliable (Fork (Leaf m1) (Fork (Leaf m3) (Leaf m2))) = False
```

(20 marks)

7. Given a tree representing a sequence of matrix multiplications, we can compute the number of integer multiplications involved. For example, given m1, m2 and m3 with dimensions as given in the previous problem, here is the number of integer multiplications for some example multiplication sequences.

```
(m1 * m2) * m3        --     2*5*3 + 2*3*2 = 42
m1 * (m2 * m3)        --     2*5*2 + 5*3*2 = 50
```

Define a function `intMults :: BinTree Matrix -> Int` that computes the number of integer multiplications in the sequence of matrix multiplications represented by the input tree `t`. (We can assume that `multipliable t` evaluates to `True`.)

Sample cases:

```
intMults (Leaf m3)                                      = 0
intMults (Fork (Fork (Leaf m1) (Leaf m2)) (Leaf m3))   = 42
intMults (Fork (Leaf m1) (Fork (Leaf m2) (Leaf m3)))   = 50
```

(20 marks)