Functional Programming

Home Assignment 3

Due: 7 May 2022 - 23:59

Instructions

- Please create a source file called ***hw3.hs*** and put all the answers there.
  The file should start with a comment, which contains your **full name** (in English) and **ID**

  -- Montgomery Burns
  -- 15926535

- **Important:** Please add the following line after the two comments:

  module HW3 where

  This line helps us to test your code.

- When writing a function - write both the **type** and the **body** of the function.
- Be sure to write functions with **exactly the specified name** (and **type signature** - if it is provided) for each exercise. You may create additional auxiliary/helper functions with whatever names and type signatures you wish.
- Try to write **small functions**, which perform just **a single task**, and then **combine** them to create more complex functions.

Exercises

Question 1

In the first question, we use the following type definition

```
type Matrix t = [[t]]
```

1.a Write a function is_square, which takes a Matrix t and determines whether it represents a square matrix.

is_square [[1,2],[3]] should return False

is_square [[1,2],[3,2]] should return True

1.b Write a function map_matrix, which takes a Matrix t, and a function and returns a new Matrix t. The elements of the new matrix are created by applying the function to each of the items in the argument.

map_matrix (^2) [[1,2],[3,2]] should return [[1,4],[9,4]]

1.c Write a function map_matrix2, which takes a Matrix t, and a function and returns a new Matrix t. The elements of the new matrix are created by applying the function to each of the items in the argument. The difference from map_matrix is that the function that we apply to each element takes not only the item but also a tuple that contains its indices. Here is the declaration for map_matrix2:

```
map_matrix2 :: (t -> (Int,Int) -> u )->Matrix t->Matrix u
```

Example

```
map_matrix2 (\item (i,j) -> (i+j)>item)   [[2,1],[1,0]]
should return [[False,False],[False,True]]
```

1.d – Write a function transpose, which takes a square matrix and transposes it (rotates values around the main axis). This function **must** use map_matrix2.

Question 2

In this question we use the following definitions

```
data BinTree = Empty | Leaf Int | Node BinTree Int BinTree
                    deriving (Show,Eq)
```

The first data type defines a simple tree containing Int values. Such a tree can be represented as a string. For example:

Node (Leaf 4) 3 (Node Empty 3 (Node (Leaf 1) 2 (Leaf 2))) can be represented as "3(4,3(,2(1,2)))"

Important:

1. We use only **positive** integers in this question
2. The string representation might contain blanks that do not alter its value. For example “1(, 2) “ is the same as “1(,2)”.

2.a

Write a function to_string, which takes a BinTree and returns its string representation as described above.

2.b

Write a function tokenize, which takes a string representing a BinTree and returns a Maybe list of its tokens. The possible tokens are taken from the following type definition. The function should return Nothing if the given string cannot be tokenized.

```
data BinTreeTok = LPar | RPar | Comma
    | Value Int deriving (Show,Eq)
```

Examples

tokenize "1(,3(,))" should return Just [Value 1,LPar,Comma,Value 3,LPar,Comma,RPar,RPar]

tokenize "1(,3(,x))" should return Nothing

2.c

Write a function compile, which takes a valid representation string and returns the BinTree it represents.