Project 3: Haskell Programming

Due Date: 11/9 by 11:59p

Important Reminder: As per the course Academic Honesty Statement, cheating of any kind will minimally result in receiving an F letter grade for the entire course.

Aims of This Project

The aims of this project are as follows:

- To give you an introduction to programming in Haskell.
- To expose you programming with high-order functions.
- To get you to use lazy evaluation and infinite data structures.

Project Specification

You are required to submit a `prj3.tar.gz` archive which contains a file `prj3-sol.hs` which can be run using the ghci implementation of Haskell on `remote.cs`.

The file should implement the functions specified in the skeleton file provided.

- The file may not include any other top-level definitions.
- If the specification says that recursion is not allowed, then your implementation must not directly use recursion.

Example Log

The following provides an edited log of interaction with the code submitted with this project:

```
$ ghci
GHCi, version 7.4.1: http://www.haskell.org/ghc/  :? for help
Loading package ghc-prim ... linking ... done.
Loading package integer-gmp ... linking ... done.
Loading package base ... linking ... done.
Prelude> :l "prj3-sol.hs"
[1 of 1] Compiling Main             ( prj3-sol.hs, interpreted )
Ok, modules loaded: Main.
*Main> quadraticRoots 2 5 2
(-0.5,-2.0)
*Main> quadraticRoots 5 6 1
(-0.2,-1.0)
*Main> take 10 $ iterateFunction (\x->x+1) 0
[0,1,2,3,4,5,6,7,8,9]
*Main> take 5 $ iterateFunction (\x->x*x) 2
[2,4,16,256,65536]
```

```
*Main> take 10 $ multiples 3
[0,3,6,9,12,15,18,21,24,27]
*Main> take 10 $ multiples (-3)
[0,-3,-6,-9,-12,-15,-18,-21,-24,-27]
*Main> take 15 $ hailstones 3
[3,10,5,16,8,4,2,1,4,2,1,4,2,1,4]
*Main> take 15 $ hailstones 7
[7,22,11,34,17,52,26,13,40,20,10,5,16,8,4]
*Main> hailstonesLen 3
8
*Main> hailstonesLen 7
17
*Main> hailstonesLen 77031
351
*Main> sumAbsDiffs []
0
*Main> sumAbsDiffs [2]
0
*Main> sumAbsDiffs [1..5]
4
*Main> sumAbsDiffs [1, 3, -5, 5]
20
*Main> distances (0, 0) []
[]
*Main> distances (0, 0) [(1, 1), (0, 2), (3, 4)]
[1.4142135623730951,2.0,5.0]
*Main> sumLengths []
0.0
*Main> sumLengths [(0, 0)]
0.0
*Main> sumLengths [(0, 0), (0, 2), (3, 6)]
7.0
*Main> sumLengths [(0, 0), (0, 2), (3, 6), (0, 2)]
12.0
*Main> occurrences "twas brillig and the slithy toves" 't'
[0,17,24,28]
*Main> occurrences "twas brillig and the slithy toves" 'x'
[]
*Main> foldTree (\t1 t t2->t1 + 3*t + t2) (\x->x*2) (Leaf 5)
10
*Main> foldTree (\t1 t t2->t1 + 3*t + t2) (\x->x*2) (Tree (Leaf 3) 2 (Leaf 4))
20
##Typed on single line
*Main> foldTree (\t1 t t2->t1 + 3*t + t2) (\x->x*2)
          (Tree (Leaf 5) 3 (Tree (Leaf 3) 2 (Leaf 4)))
39
*Main> flattenTree (Leaf 5)
[5]
*Main> flattenTree  (Tree (Leaf 5) 3 (Tree (Leaf 3) 2 (Leaf 4)))
[5,3,3,2,4]
##Typed on single line
*Main> flattenTree  (Tree (Leaf [5]) [3]
                          (Tree (Leaf [3, 2]) [1, 2] (Leaf [4, 5])))
[[5],[3],[3,2],[1,2],[4,5]]
##Typed on single line
*Main> catenateTreeLists  (Tree (Leaf [5]) [3]
                                (Tree (Leaf [3, 2]) [1, 2] (Leaf [4, 5])))
```

```
[5,3,3,2,1,2,4,5]
##Typed on single line
*Main> catenateTreeLists  (Tree (Leaf "twas ") "brillig "
                           (Tree (Leaf "and ") "the slithy " (Leaf "toves")))
"twas brillig and the slithy toves"
*Main> :quit
Leaving GHCi.
$
```

Provided Files

The ./files directory contains the following:

Makefile
: This file contains a `submit` target such that typing `make submit` will create a `prj3.tar.gz` archive containing the files to be submitted. The `clean` target will remove the archive.

  You may edit this file if you choose to use a different organization for your project. When editing, watch out for tabs (the first character of any command-line **must be a tab character**).

README
: A template README; replace the `XXX` with your name, B-number and email. You may add any other information you believe is relevant to your project submission. In particular, you should document the data-structure used for your word-store.

prj3-sol.hs
: A file containing skeletons functions for the exercises.

Hints

You may choose to follow the following hints (they are not by any means required). They assume that you are using the project structure supported by the provided Makefile,.

- The solutions to many of the exercises easily fit on a single line.
- Most of the exercises require the use of higher-order functions.
- Some of the exercises can avoid the use of recursion by using functions like `map`, `foldl`, `foldr`, or `filter`.
- You may use the ghci debugger; alternately, you may simply use the trace https://hackage.haskell.org/package/base-4.9.0.0/docs/Debug-Trace.html function.

Submission

You will need to submit a compressed archive file `prj3.tar.gz` which contains all the files necessary to build your jar file. Additionally, this archive **must** contain a `README` file which should minimally contain your name, B-number, email, the status of your project and any other information you believe is

relevant.

If you are using the suggested project structure, then the provided Makefile provides a `submit` target which will build the compressed archive for you; simply type `make submit`.

Note that it is your responsibility to ensure that your submission is complete. To test whether your archive is complete, simply unpack it into a empty directory and see if it runs correctly.

To submit the above archive, please use blackboard by following the `Content->Projects` link.