1 Project 4

Due Date: 5/3 before midnight.

Important Reminder: As per the course *Academic Honesty Statement*, cheating of any kind will minimally result in your letter grade for the entire course being reduced by one level.

1.1 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.

1.2 Project Specification

Update your github repository with a directory submit/prj4-sol to contain a file `prj4-sol.hs` which implement the functions specified in the skeleton file prj4-sol.hs:

- 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.

The file LOG provides an annotated sample log of the operation of these functions.

1.3 Provided Files

The <./prj4-sol> directory contains the following:

prj4-sol.hs A file containing the specifications for the exercises. You should use this as a starting point for your work.

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.

1.4 Hints

- 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`.
- Even though the requirements preclude the use of auxiliary top-level definitions, this is not really a restriction since functions can contain internal definitions using `let` or `where`.
- See Debugging and Haskell/Debugging for tips on how to debug Haskell programs.