Assignment 2

Create a Haskell script file named `stringsTable.hs` to submit your work for this assignment. Use your hash table data structure from Lab 3.

---

1) Edit your code with use of the `type` keyword

This is to make your code for your hash table easier to read. We saw examples of this with the phoneBook for the association lists and Map data types. Edit your code so any nested types can be stored inside your hash table (like a type parameter we have seen typically labelled with `a`, `b`, `c`, …). Maintain your hash function to keep working with strings. The goal here is to operate on the nested string elements that might result in other types stored in their place. If you already have the above functionality, then you are already ahead.

2) Implement your hash table as an instance of `Functor`

- make a call to fmap with a function that
  - calculates the length of every string in a demonstration of your hash table
  - with 50 random input strings from the following webpage:

    https://www.randomlists.com/random-words?dup=false&qty=50

> Be concise about how you insert the words into an example table. You will not earn marks for calling your function repeatedly, nor recursively. We have learned how to iteratively apply your insert function with one line of code.
>
> (there are a few ads, but the set of words are easy to cut-and-paste)

- your `fmap` call will return a hash table of integers
  (do not rehash the integer elements, leave each element in its place after the `fmap`)
  - make sure your hash table is deriving `Show`
  - store the result in a variable called `lengthsTable`

(third part on next page)

3) Implement your hash table as an instance of `Foldable`

- demonstrate the use of your hash table as a foldable type by using foldMap
  - use with the `Sum` monoid discussed in Chapter 12
  - on the result of your `fmap` stored in `lengthsTable`
  - to obtain the total number of characters of all words stored in your table
  - and store the resulting total in a variable called `totalCharCount`

No marks for calculating total character count in other ways, as the purpose of this assignment is to implement `Functor` and `Foldable` on your hash table data structure.

Keep in mind that an instance of `Foldable` itself does not need to be a `Monoid` but can work with monoids to help process nested elements inside whatever is foldable.

If you implemented a hash table dealing with collisions using probing, then making your table an instance of `Functor` and `Foldable` should be straightforward. Otherwise, as a hint, you will need to apply nested map and fold functions for dealing with nested lists. Make sure to import the necessary modules in your script.

Rubric

10 marks total

- [2 mark] have type parameter for nested elements of hash table
- [2 marks] implement your hash table as a `Functor` instance
- [1 mark] demonstrate use of `fmap`
- [3 marks] implement your hash table as a `Foldable` instance
- [2 marks] demonstrate use of `foldMap` and `Sum` monoid

Submit `stringsTable.hs` before Nov 5 end of day.