Assignment #6
CS 3060 Programming Languages, Fall 2023
Instructor: S. Roy

Haskell #2

Due Date: Nov 29 @ 11:59 PM.
Total Points: 60 points

Directions: Using the source provided via Gitlab `https://gitlab.com/sanroy/fa23-cs3060-hw/`, complete the assignment below. The process for completing this assignment should be as follows:

1. You already forked the Repository "sanroy/fa23-cs3060-hw" to a repository "yourId/fa23-cs3060-hw" under your username. If not, do it now.
2. Get a copy of hw6 folder in "sanroy/fa23-cs3060-hw" repository as a hw6 folder in your repository "yourId/fa23-cs3060-hw"
3. Complete the assignment, committing changes to git. Each task code should be in a separate file. As an example, task1a.hs for Task 1a.
4. Push all commits to your Gitlab repository
5. If you have done yet done so, add TA and Roy as a member of your Gitlab repository

Tasks:

1. **Task #1: (25 points)** Write a Haskell function for each of the following. In your code, you need to specify the **input and output type** of each function.

   (a) (10 points) Write a Haskell function *bar* which takes three integers *x, y, z* as input parameters and returns a tuple, (*u, v*) where *u* is the sum of all odd integers bigger than *x* and smaller than *y*, and *v* is the sum of all even integers bigger than *y* and smaller than *z*. You need to use *foldl* to compute the above sums. As an example, if *x* is 2, *y* is 20, and *z* is 40, then *bar* will compute $(3+5+\ldots+19, 22+24+\ldots+38)$. Note that *x, y, z* can be any integer, positive, zero, or negative. *Writing README carries 1 point.*

   (b)(8 points) Write a Haskell function *digitCount* which takes an integer *num* as input, and counts how many digits (say *c*1) in *num* are odd (i.e., 1, 3, 5, 7, or 9) and how many digits (say *c*2) in *num* are even (i.e., 0, 2, 4, 6, or 8), and returns two counts (*c*1, *c*2) as a single tuple. As an example, if *num* is 12364, then *digitCount* returns (2,3). As an example, if *num* is -136, then *digitCount* returns (2,1). *Writing README carries 1 point.*

   (c)(7 points) Write a Haskell function *foo* which takes a list of numbers as input, and counts how many numbers have less than 5 digits and more than 2 digits, and returns the count. As an example, if input *list* is [123, 56, -78812], then *foo* returns 1. *Writing README carries 1 point.*

2. **Task #2: (15 points)** Refer to the user-defined types `Card` and `Hand` in the textbook (cards-with-show.hs). Also, see the `value` and `cardvalue` function therein. Write a Haskell function for each of the following. In your code, you need to specify the **input and output type** of each function.

   (a) (4 points) Write a function named `biggestCard` which takes three Cards and returns the highest value Card. If there is a tie, then any candidate Card can be returned.

   (b) (4 points) Write a function named `sumValue` which takes a Hand and returns the sum of all values of cards in that Hand.

   (c) (7 points) Write a function named `biggestHand` which takes a list of Hands and returns the Hand which has the highest `average value`. If there is a tie, then any candidate Hand can be returned.

   Writing README carries 2 points.

3. **Task #3: (13 points)** Say we are interested in songs.

   (a) (4 points) Assume that a book has three attributes: (i) title of the book which is a string, (ii) price of the book, which is a decimal number, and (iii) author name which is a string. Define a data type named Book. Write code to create the following books: ("Great day", 290.2, "PC") which we call book1, ("To kill a mocking bird", 310.4, "HL") which we call book2, ("Fermat enigma", 297.3, "MS") which we call book3.

   (b) (4 points) Consider that you have a collection of books. Define a data type named BookColln. Write code to create an collection which has the above three books.

   (c) (5 points) Write a function named `contains` which takes two parameters: a book b and a collection x . The function returns a boolean depending on whether collection x contains book b. In your code, you need to specify the **input and output type** of the function. Test the function to search book2 in the collection that you created before. Your function should also work for any other book and another collection.

4. **Task #4: (7 points)** Read the attached haskell file, process-story.hs, which is supposed to report the number of words in story1.txt (also attached). Most of the code is written for you. You have to complete the code. Note that you have to use the given code to do this task.