Assignment #8
CS 3060 Programming Languages, Fall 2020
Instructor: S. Roy

Haskell #2

Due Date: Dec 1 @ 11:59 PM.
Total Points: 60 points

Directions: Using the source provided via Gitlab `https://gitlab.com/sanroy/fa20-cs3060-hw/`, complete the assignment below. The process for completing this assignment should be as follows:

1. You already forked the Repository "sanroy/fa20-cs3060-hw" to a repository "yourId/fa20-cs3060-hw" under your username. If not, do it now.
2. Get a copy of hw8 folder in "sanroy/fa20-cs3060-hw" repository as a hw8 folder in your repository "yourId/fa20-cs3060-hw"
3. Complete the assignment, committing changes to git. Each task code should be in a separate file. As an example, task1.hs for Task 1.
4. Push all commits to your Gitlab repository
5. If you have done yet done so, sdd TA (username: prabeshpaudel) as a member of your Gitlab repository

Tasks:

1. **Task #1: (30 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 an integer x as input parameter and returns the product of all positive even integer(y)s' cubes whereas y is smaller than x. You need to use *foldl* to do the above computation. As an example, if x is 10, then *bar* will compute $(2^3 \times 4^3 \times \ldots \times 8^3)$. *Writing README carries 1 point.*

   (b)(10 points) Write a Haskell function *upperCharCount* which takes a string *word* as input, and counts how many letters in *word* are in uppercase, and returns the count. As an example, if *word* is "abDfGi", then *upperCharCount* returns 2. *Writing README carries 1 point.*

   (c)(10 points) Write a Haskell function *shrtStrCount* which takes a list of strings as input, and counts how many strings have length less than 3, and returns the count. As an example, if input *list* is ["abcd", "de", "fghtestwsd"], then *shrtStrCount* returns 1. *Writing README carries 1 point.*

2. **Task #2: (30 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) (10 points) Write a function named `upperCard` which takes two Cards and returns the higher value Card. If there is a tie, then either Card can be returned.

   (b) (10 points) Write a function named `sumValue` which takes a Hand and returns the sum of all values of cards in that hand.

   (c) (10 points) Write a function named `lowerHand` which takes two Hands and returns the Hand which has the lower value. If there is a tie, then either Hand can be returned.

   Writing README carries 2 points.