Assignment #9
CS 3060 Programming Languages, Spring 2021
Instructor: Dr. Roy

Scala-Haskell

Due Date: Apr 24 (Sat) @ 11.59 PM.
Total Points: 50 points
Note: This is an extra assignment, which may help you improve your grade as we will be picking up your best 8 assignments out of 9 assignments.

Directions: Using the source provided via Gitlab `https://gitlab.com/sanroy/sp21-cs3060-hw/`, complete the assignment below. The process for completing this assignment should be as follows:

1. You already forked the Repository "sanroy/sp21-cs3060-hw" to a repository "yourId/sp21-cs3060-hw" under your username. If not, do it now.
2. Get a copy of hw9 folder in "sanroy/sp21-cs3060-hw" repository as a hw9 folder in your repository "yourId/sp21-cs3060-hw"
3. Complete the assignment, committing changes to git. Each task code should be in a separate file. As an example, task1.scala 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: (10 points)** Scala. In Task 6 of HW5 (Scala) we wrote a Scala program to process two textfiles (i.e. two stories) that you downloaded from a website (http://www.textfiles.com/stories/). The Scala program was to compute some statistics of those two files (e.g. word count, etc.). I guess you will find it even more exciting if we can automatically grab all story files (maybe, more than 100 stories it has) that appear on the same website and process them to collect statistics. That is exactly what you will do in the current task. In particular, write a Scala program to process all stories (whereas each story is a text file) present on http://www.textfiles.com/stories/ and report the average number of words in a story.

   Hint: In HW6 we learned how to programmatically find all links on a webpage. Here each link corresponds to one story textfile.

2. **Task #2: (6 points)** Scala. Let's get back to Task 2 of HW6 (Scala) where we developed a mini web crawler. You know that in Task 2c we did not use parallel computation (i.e. multi-Threaded processing) last time. Your current task is to do exactly that, i.e., redo Task 2c through parallel computation. You need to use the concept of `par` collection, and concept of functional programming while ensuring that your program is free from `side effect`. *Writing README carries 1 point.*

3. **Task #3a: (10 points)** Haskell. Write a Haskell function named `myCount` which takes two parameters, an item `p` and a `list`. The myCount function returns how many times `p` appears in `list`. Your function needs to be generic, e.g. should work for integer, characters or anything else. Test your function as follows and report the result in README.

   myCount 'a' ['a','b','a','b','a'] should return 3

   myCount 3 [5,1,5,2,3] should return 1

   Task #3b: (3 points) Test your function on a list of Cards. The Card type is defined in Haskell lecture ppt 2. Report the result in README.

   Task #3c: (3 points) Test your function on a list of Books. The Book type is defined in Haskell group activity 15. Report the result in README.

   Writing README carries 2 points.

4. **Task #4: (18 points)** Haskell. Haskell recursive type: Write a function `greaterThan10` which takes a binary tree as input and returns the number of nodes (in the tree) that hold value greater than 10. Use the following data type for the binary tree, where each internal node as well as each leaf node holds a value.

   data Tree a = Leaf a | BinaryTree (Tree a) a (Tree a) deriving (Show)

   Build a tree of 12 or more nodes with height 3 or more (where each node holding an integer) and apply your `greaterThan10` function on the tree, and report the output in README file. *Writing README carries 2 points.*