CS1083 Assignment #11 – Winter 2021

Due: **MONDAY, April 12th before 4:00pm** (Atlantic), submitted in the Assignment 11 dropbox in Desire2Learn. (Read the submission instructions at the end of this document carefully).

The purpose of this assignment is to provide you with practice with binary search trees.

This assignment is to be done individually. If you have questions, direct them to a tutor/assistant during a help session in the "Faculty of Computer Science Student Success Centre" team or to your course instructor.

Name Counts

For this assignment you will be counting names. The number of occurrences of each name appearing in a class list is stored in a binary search tree where each pair is the name and its count. For the list of names:

```
Elizabeth Anne Flynn
Sarah Blair
Abe Corey
Blair Elliot
COREY FLYNN
anne blair
```

The following name and count pairs would be:

```
(abe, 1)
(anne, 2)
(blair, 3)
(corey, 2)
(elizabeth, 1)
(elliot, 1)
(flynn, 2)
(sarah, 1)
```

You will be generating the pairs by reading a list of names from a text file. Before starting the assignment, draw the binary search tree that is generated by reading in the above names, where the tree is ordered alphabetically by name.

Write a class called **NameCountTree.java** that contains a definition for a binary search tree of Strings. The class must contain two inner class, Pair and Node.

The **Pair** inner class is used to store the String value and count and must override the toString method to print out the name, count pair in the format: **`(sarah, 1)`**

The **Node** inner class holds information for a single node in the tree (a node has a String value and a count, stored in an instance of the **Pair** inner class, and references to the left and right children).

Along with a constructor, your implementation of the **NameCountTree** must also contain the following 5 methods:

- `add(String valueIn)`: public method that calls the recursive add method.
- `add(String valueIn, Node root)`: private **recursive** method that increments the count of String `valueIn` by 1, or adds the String `valueIn` to the tree with a count of 1.
- `readText(String filename)`: Reads a text file, parses each String, and counts all the names (using the lower case spelling of the name) by adding them to a binary search tree of String/count pairs (make use of the add method).
- `print()`: public method that calls the recursive `print` method.
- `print(Node root)`: private **recursive** method that prints the nodes of the tree in alphabetical order.
- `printMin()`: prints the String/count pair of the name with the alphabetically lowest value (ie: **`(abe, 1)`**for the example above).

Note that String/count pairs can be printed by making use of the Pair class's toString() method. A driver **NameCountDriver.java** file has been provided for you in D2L, which reads a text file, and prints the name counts in order and then prints the minimum entry in the tree.

Test your program by creating a test case called **test1.in** containing the names:

```
Elizabeth Anne Flynn
Sarah Blair
Abe Corey
Blair Elliot
COREY FLYNN
anne blair
```

Create **two other test case** that you make up.

Code must be appropriately commented (including **Javadoc comments**).

For this assignment, only an electronic submission is required.
Your electronic submission (submitted via Desire2Learn) will consist of two files:

i. a written report. This should begin with a title page that includes: the course (CS 1083), your section (FR01B, FR02B, FR03B), the assignment number, your full name, and your UNB student number. That should be followed by two sections, with each part clearly identified with a section heading. Include:

- a single pdf file containing a listing of the source code for the **NameCountTree** class, the 3 input files, and the output of the driver for the 3 test cases captured from the terminal window.

This written report should be prepared using a word processor; we recommend using Microsoft Word (i.e. create a .docx file for your report). Copy & paste your java source code and the contents of the input files, and insert images of your terminal window containing the output into the report document. Add appropriate headings for each part. Fix up the formatting where necessary, adjusting line breaks & page breaks to ensure that your document is easy to read. Use a monospaced font for your code to maintain proper indentation.) Once the report is complete and you've checked it all over, save the .docx file for your own records, and then **save a second copy in pdf format for submission**. (Note: Be sure to open that file in a pdf viewer to verify that the pdf was generated correctly.) **The SINGLE pdf file containing your report will be submitted to the appropriate assignment drop box on Desire2Learn**. (It is important that you submit a pdf file and NOT the original Word document. This pdf will allow the marker to write comments directly on your work to give you better feedback.)

Note: Please name this report as follows: **YourName_A11_Report.pdf**

ii. an archive file (.zip) that contains all your work for this assignment. Make sure that your archive includes **all source code** (all .java files and test case files in case the marker wishes to compile & run your code). This archive should be submitted as a single file to the appropriate drop box on Desire2Learn. Note: Please name this archive file as follows: **YourName_A11_Archive.zip**