**FINAL (Feb/22)**

**Assignment 2**

In this assignment, you will need to make use the full dictionary file:

* [Dictionary text file](http://leda.science.uoit.ca/teaching/sysdev/assignments/a1/helper/dict.txt)

**0.0**

**(1) Counting characters:**

In this section, you are to experiment with the performance differences in buffered versus simple character I/O.

**Problem:** Compute the frequency of the alphabet A -- Z appearing as **the last character of a word** over the entire dictionary, and display the letters with their frequencies in no particular order.

**Perform case insensitive counting.** So, a and A are both treated as the same letter A.

(1.a) Unbuffered character stream:

* Construct a Reader **without** buffering.
* Build an executable class UnbufCount which displays
  1. the frequences of A to Z in no particular order.
  2. the total time it took to complete the frequency counting.

The following is the usage and

$ java UnbufCount dict.txt

Took: 230 ms

D: 40293

Z: 302

X: 34

A: 29345

...

(1.b) Buffered stream:

* Construct a BufferedReader with the specified buffer size.
* Build an executable class BufCount which displays the same information as UnbufCount. However it must accept two parameters: dictionary file name, and the size of the buffer. The implementation should use readLine to make the code much easier (compared to (1.a)).
* $ java BufCount dict.txt 100
* Took: 22 ms
* D: 40293
* Z: 302
* X: 34
* A: 29345

...

* Plot the performance of BufCount with varying buffer size ranging from 1 to 1000.
  + Include the plot in your report, **and**
  + Post the plot on course G+ community.
* Repeat the performance of BufCount on a USB drive.

**1.0**

**Submission**

Your submission should include files:

.

├── BufCount.java

├── report.pdf <- includes the performance plot

└── UnbufCount.java

Grading scheme:

* (1.a)
* Coding : 10
* Functional correctness : 30
* (1.b)
* Coding : 10
* Functional correctness : 30
* Report
* Plots : 10
* Discussion : 10

**2.0**