CMPS 12B-02, Fall 2017
HW5: All’s Will that Hashes Will

Due: Tuesday, Dec 5, 2017 by 11:59pm

- All assignments must be submitted through git. Please look at the Piazza guide on submitting assignments.
- Please follow instructions properly, include naming conventions, and carefully read through the input/output formats. There’s no spec file associated with HW5, so you won’t have to run the checking script. However, you’re still responsible for submitting all the files that are required by this homework.
- If your HW5 solution is submitted by Tuesday, December 5, 2017 by 11:59pm, then it may get full credit. One point will be deducted for late HW4 solutions that are submitted by Wednesday, December 6 by 11:59pm. Solutions submitted after that will not be graded. Don’t forget to fill in your commit id in the Google Form.
- Clearly acknowledge sources in a README file, and indicate if you discussed the problems with other students or groups. Please submit a README file even if you didn’t use any sources. In all cases, the course policy on collaboration applies, and you should refrain from getting direct answers from anybody or any source. If in doubt, please ask the Instructor or the TAs.

1 Problem description

Main objective: You’re going to do a word analysis of Shakespeare’s works. The full text of all the compositions of William Shakespeare is in the file Shakespeare.txt, which is in Resources→HW5 on Piazza. The file is courtesy of Project Gutenberg, so it contains some copyright messages.

Write a Java program that can answer the following queries about the entire contents of that file.

- For any word, how many times did Shakespeare use that word?
- For any word length *L* and number *K*, what are the *K* most frequent words that have length *L*?

For HW5 only, you may use any data structures that you want, including built-in Java types/classes. But as usual, you must do the assignment yourself.

When analyzing Shakespeare.txt file, assume that a "word" is a sequence of characters between 'a' and 'z' or 'A' and 'Z'. All other characters are "word separators", and should not be attributed to any word. Word counting should be case sensitive, i.e., "Shakespeare" and "sHaKeSpEaRe" should be counted as different words. You can use the compareTo operation to compare Java Strings. The Shakespeare.txt file contains some extraneous information (e.g., copyright notices) that's not part of Shakespeare's works. Don't bother to delete this, since it will have a negligible effect on the output.

Format: You should provide a Makefile. On running `make`, it should create Bard.jar. There is no argument. The code should read in a file *input.txt*, where each line contains a different query. The query is either a word, or a pair of numbers. For example:

```
Brutus
Caliban
9 1
7 2
yonder
10 3
```

If the query is a word (such as "Brutus" or "yonder"), then you should output the word and the number of times that the word appears in the text. For example, if Brutus appears 206 times, you would output "Brutus 206", without the quotes. If the query is a number pair, such as 7 2, then you should just output (on the same line, separated by a space) the 2 most frequent words that have length 7. (Don't output the 7 and the 2.) If there are ties, then you may output any appropriate words.

The output should be in the file *analysis.txt*, where line j of the file is the answer for the query in line j of *input.txt*

2 Grading

Your code should terminate within 5 minutes when run on the example *input.txt* shown above. If it doesn't, we will not give you credit. (You will not be given the corresponding *analysis.txt* output, and we may run your code on other *input.txt* files.) No spec file for checker is provided for HW5, but you should still be sure to submit (via your GitLab repository) your java files, your Makefile (which will create the Bard.jar file, as described under **Format** above) and your README file. To receive credit, you must also submit your git commit id using the Google Form for HW5.

1. (10 points) For a full solution as described above.
2. (8 points) Only answers word frequency queries correctly.
3. (8 points) Only answers number pair queries correctly.