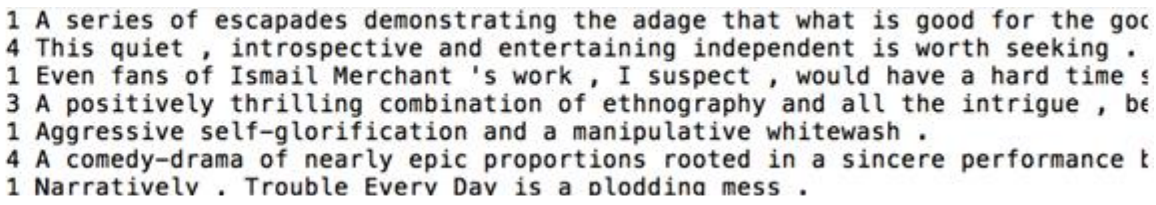
CSCI 2073 – Fall 2017 – Programming Assignment 4
Movie Review Analysis

Your task is to write a program that uses a collection of existing movie reviews to assign sentiment scores to additional reviews provided as input. The program is an example of *Machine Learning*, a branch of computer science that studies algorithms that build models from sample inputs and use those models to make predictions or decisions.

The data that the algorithm is going to "learn" from is a set of movie reviews extracted from the *Rotten Tomatoes* database, in which the sentiment of each review has been manually rated on a scale from 0 to 4. The labels are:

0 - negative
1 - somewhat negative
2 - neutral
3 - somewhat positive
4 - positive

The data has been formatted so that it is relatively easy for a program to process: each line contains an integer in the range 0-4 indicating the manually assigned score for the review, followed by a sequence of words -separated by spaces- comprising the text of a review. Punctuation symbols and abbreviations involving apostrophes have been conveniently tokenized. A sample segment of a data file is:



```
1 A series of escapades demonstrating the adage that what is good for the goc
4 This quiet , introspective and entertaining independent is worth seeking .
1 Even fans of Ismail Merchant 's work , I suspect , would have a hard time s
3 A positively thrilling combination of ethnography and all the intrigue , be
1 Aggressive self-glorification and a manipulative whitewash .
4 A comedy-drama of nearly epic proportions rooted in a sincere performance t
1 Narratively . Trouble Every Day is a plodding mess .
```

The basic idea is to build a collection of words and associated scores from the data. Each word is given the score of the line in which it appears. Since a word will normally occur several times, its score will be the average of the scores associated with all its occurrences. For instance, from the segment above, the word *epic* would be associated with a score of 4, while the word *the* would be associated with a score of 1.67 (the average of 1, 1, and 3).

To make the scores more meaningful, your program should ignore case when processing and comparing words. It should also ignore any punctuation symbols, two-letter words, subject pronouns (*I, you, it, he, she, we, they*), articles (*a, an, the*), and conjunctions (*and, or, but*). Once the collection of words and scores is built from the training set, your program will assign scores to additional reviews by taking the average score of the relevant words in the review.

Your solution will be stored in the class `MovieReviews`, which will have the following required methods (you may have additional helper methods to accomplish the task at hand):

- `MovieReviews(String filename, int numberOfLines):` a constructor that builds the word collection using at most the given number of lines from the input file.
- `double wordScore(String word):` returns the score associated with a word in the collection. If the word is not in the collection, the neutral score (2.0) is returned.
- `double reviewScore(String review):` returns the calculated score for a new review given as argument (the average score for the relevant words in the review).
- `String mostPositive():` returns the word with the highest score in the collection, provided the word occurs at least twice.
- `String mostNegative():` returns the word with the lowest score in the collection, provided the word occurs at least twice.

The ReviewTester.java program as well as a sample data file have been provided to help you test your solution. As always, you should test your solution thoroughly before submitting it to Mimir for final testing. When finished, submit MovieReviews.java (and any other .java files you developed).