TI2736-B: Assignment 2
Big Data Processing

Due date: 02.12.2015 @ 11.59pm

Please submit your report and code via Blackboard (please do **not** copy & paste your code directly into the report). Make sure to include your name and student number in your report.
This time, the assignment focuses on Hadoop. The goal is to make you comfortable with writing and designing basic Hadoop programs. For exercises 1-3, submit the source code as well as the statistics/results you gathered when running the Hadoop job on our `shakespeare` corpus. For exercises 4 and 5, submit pseudo-code.
When writing your (pseudo-)code, keep in mind that your code needs to be scalable, i.e. the code should also be able to run unchanged on a Terabyte corpus.

1. In Assignment 2, you were asked to run `WordCount.java`. Modify the program again to output the following statistics over the `shakespeare` corpus (the same corpus as in Assignment 2):

   - the number of unique (or distinct) terms in the corpus
   - the number of words in the corpus that start with the letter T/t
   - the number of terms appearing less than 5 times
   - the number of files read overall
   - the 5 most often occurring terms and their frequency in the corpus

   All these statistics should be computed within a single Hadoop job.

   You can use any of the devices we covered in the previous lectures (e.g. Counters, Partitioners, Combiners, setup/clean, etc.). The **output** of the job in the end should be the five terms that occur most frequently in the corpus together with their actual frequency. Submit the source code and the job output. Report the numbers you got for each of the queries listed above.

2. Starting off with the original `WordCount.java` again, write a Hadoop job that computes the number of terms appearing in only one document of the corpus. A term may appear multiple times in a single document. Only if the term does not appear in any other document of the corpus, should it be included in the count. Submit your source code and report how many terms appear only in one document of our `shakespeare` corpus.

3. Starting off with the original `WordCount.java` again, write a Hadoop job that computes for each of the three stopwords **the**, **of** and **and** the five terms that occur most often directly after these terms within our `shakespeare` corpus. As a concrete example, consider this toy corpus consisting of two documents where we want to determine the most often occurring terms after the stopword **the**:

   - *The dog walks around and the cat of the lady generally likes dogs.*
   - *The cat of the lady and the dog of the lady hate each other.*

   Tthe three terms most often occurring directly after **the** are *lady* (3 times), *cat* (2 times) and *dog* (2 times).

   All counts should be computed within a single Hadoop job. The output of this program should be 15 lines of the form *[stopword] [term] [count]*.

4. In this exercise, a data set is described and a number of queries are listed. Imagine you want to process the data set on Hadoop. Write out in **pseudo-code** the steps taken in the Mapper and Reducer (the level of detail should be similar to the pseudo-code in the lectures) to answer the different queries listed below.

   The data set contains weather information and looks as follows:

```
Paris/France, Dec, 2009, 2.34
Paris/France, Jan, 2010, 2.52
..
London/UK, Dec, 1987, 1.4
...
London/UK, Nov, 2013, 2.0
```

   Each line has the same setup: location (city), date given as month and year, and finally the average temperature of that month in that year[1]. Assume that in each `map()` call one line of text is being processed. The data set contains data from 1950 until 2013 and for some locations the weather data is incomplete (no entry for a few months/years).

   Note that you can write one job per wanted information or a job that retrieves several items within a single job. Write pseudo-code to retrieve as output of the Reducer or in a Counter:

   - the minimum temperature ever recorded for each location
   - the minimum temperature overall in the data set
   - for each location, the average temperature per month for all years recorded

[1] Concretely, considering line 1: in December 2009, the average temperature in Paris (France) was 2.34 degrees

- for each location, the years for which not a single measurement exists

5. This exercise is in the same spirit as exercise 4, just with a different data set and different queries. Provided is a data set which could be derived from a question-answer (QA) portal such as Stackoverflow:

```
1, Q, -1, "Hadoop and MapReduce - difference?", 1-1-2009, U23
2, Q, -1, "Why is there such a hype about Scala?", 2-1-2009, U43
3, A, 1, "Hadoop is the open-source implementation ...", 2-1-2009, U43
4, Q, -1, "Do lambda expressions exist in Java?", 4-1-2009, U80
5, A, 4, "Java 8 comes with them", 4-1-2009, U23
6, A, 4, "Lambda expressions are just a hype", 5-1-2009, U2
....
```

   Every line contains the following fields:

   (a) row id

   (b) indicator "Q" if it is a question and "A" if it is an answer to a question

   (c) if the entry is an answer, the question id (given as row id) the answer refers to is given; for questions this entry is -1

   (d) question/answer text

   (e) the posting time

   (f) the user id of the posting user

   As an example, user U23 posts a question (row 1), which is answered by user U43 (row 3). The question posted in row 4 is answered by user U23 in row 5 and by user U2 in row 6. Thus, several answers can be given to a question.

   Write pseudo-code to retrieve as Reducer output or Counter value:

   - the list of all questions (specifically their row IDs) that have not been answered by anybody.
   - the overall number of users as well as the number of users that have posted questions AND answers
   - the id of the question which has received the most answers.
   - the user id of the most active user (most posts, answers or questions)
   - the calendar date on which the most questions have been posted.