Upload your solutions on or before the due date (mentioned in the moodle) in a single zip file using your group id as the file name. Include some brief instructions on how to run your solution to each problem in a file called `problem_X.txt`. *Each group needs to demonstrate the submitted solutions to one of the teaching assistants on a mutually decided date. Each group may internally divide into sub-groups to focus on different problems in the assignment. Each member in the group may be evaluated independently during the demo session. All the members in a group must participate in the demo.*

MapReduce & Apache Hadoop

Problem 1. Ratio of Nouns **9 Points**

A sentence is composed of multiple words and each word has its specific grammatical category, also known as part-of-speech. Part-of-speech tagging (POS tagging) is the process of identifying the corresponding part-of-speech (verb, noun, adjective, etc.) of words in a sentence.

- From all words in the `Wikipedia-EN-20120601_ARTICLES.tar.gz` corpus, calculate the percentage of nouns in it by using two MapReduce jobs. **4 Points**
- Could you solve this exercise with a single job? Compare the running time of the solution using one iteration against the one using two. **5 Points**

Hint 1: To identify the POS tags, you may use the OpenNLP library and a pre-trained POS tag model, which are available from `opennlp-tools-1.9.3.jar` *and* `opennlp-en-ud-ewt-pos-1.0-1.9.3.bin` *on the moodle. You may check the PosTag.java file on the moodle to learn more on how to use the OpenNLP library.*

Hint 2: To chain multiple MapReduce jobs you may follow the following tutorial `https://towardsdatascience.com/chaining-multiple-mapreduce-jobs-with-hadoop-java-832a326cbfa7`.

Hint 3: To execute external libraries on Hadoop - as the `opennlp-tools-1.9.3.jar` *- you shall add the directory of those libraries to the* `HADOOP_CLASSPATH` *environment variable, as follows:* `export HADOOP_CLASSPATH="<PATH_TO_LIBRARIES>/*"`

Problem 2. Indexing Documents via Hadoop **12 Points**

1. Implement a pair of a `Map` and a `Reduce` function which, for each distinct term that occurs in any of the text documents in `Wikipedia-EN-20120601_ARTICLES.tar.gz`, counts the number of distinct documents in which the term appears. We will call this value the Document Frequency (DF) of that term in the entire set of Wikipedia articles. Store the resulting DF values of all terms in a single TSV file with the following schema:

   ```
   TERM<tab>DF
   ```

   Hint: Also here consider the Porter stemmer that is available from `opennlp-tools-1.9.3.jar` *on* *moodle. uni. lu. After importing the library with* `import opennlp.tools.stemmer.PorterStemmer;` *and creating an instance of* `PorterStemmer stemmer = new PorterStemmer();` *use* `stemmer.stem(token);` *for stemming an input token.* **6 Points**

2. Implement another pair of a `Map` and a `Reduce` function which, for each document in `WikipediaEN-20120601_ARTICLES.tar.gz`, first counts the number of occurrences of each distinct term within the given document. We will call this value the Term Frequency (TF) of that term in the given document.

   In a second step, your combined MapReduce function should multiply the TF value of each such term with the inverse of the logarithm of the normalized DF value calculated by the previous MapReduce function, i.e., SCORE = TF×log(10000/DF+1) for each combination of a term and a document. You may cache the former TSV file with the DF values by adding it via `Job.addCacheFile(<path-to-DF-file>)` in the driver function of your MapReduce program. The `Map` class should then load this file upon initialization into an appropriate main-memory data structure.

   The result of this MapReduce function should be a single TSV file that has the same schema as the file we used in the previous exercise sheets: **6 Points**

   ```
   ID<tab>TERM<tab>SCORE
   ```

   Hint: You may use either the given document URL's or simple integer id's for the `ID` *field of this TSV file (as long as they uniquely identify the original text article).*

Problem 3. Processing Joins via Hadoop **9 Points**

For this exercise, consider either the file `Wikipedia-EN-20120601_KEYWORDS.TSV.gz` provided on the moodle or your own TSV file you created for Problem 2 of this exercise sheet.

- Implement the following three "Boolean Retrieval" queries using as pairs of `Map` and `Reduce` functions that perform *reduce-side joins* over the inverted lists for the four (stemmed) terms `infantri`, `reinforc`, `brigad` and `fire`.

  1. *Query 1*: Find URLs of Wikipedia articles that contain *all* of the stemmed keywords `infantri`, `reinforc`, `brigad`, and `fire`. **3 Points**
  2. *Query 2*: Find URLs of Wikipedia articles that contain *any* of the stemmed keywords `infantri`, `reinforc`, `brigad`, or `fire`. **3 Points**
  3. *Query 3*: Find URLs of Wikipedia articles that contain the stemmed keyword `reinforc` but *not* any of the stemmed keywords `infantri`, `brigad`, or `fire`. **3 Points**

  Note: The URLs of the documents can be obtained by performing a join operation with the contents of `Wikipedia-EN20120601_REVISION_URIS.TSV.gz`.