# IS 303 – Architectural Analysis: Assignment #2

## Due: Week 7

Instructions:

* **Your report *must be A4*. The first page of your report must include your section and team number (ex: G1T1). The zip file of your code must also include your section and team number (ex: Assignment3\_G1T1\_code.zip).**
* Your team should work together. You may consult other teams or sources without penalty, but any input beyond the course materials—whether from individuals or documents — must be acknowledged in your submission.
* Most problems will require two parts: 1) some coding and 2) a report of your finding (questions to answer, screen shots, or graphs). Place your report in a Word doc or a pdf and submit via eLearn. Place your code in a zip file and submit via eLearn. The report and the code should be separate.

Marks:

* 4.5% of final grade

**Problem #1 (MapReduce)**

Use a MapReduce algorithm to count the number of times different words appear in the complete works of William Shakespeare. The output of your reducer(s) should list the words counted and the number of times that each word occurred.

Your mapper should ignore white spaces and punctuation. For example: “time.” and “time” should be counted as the same word. Your mapper should also count uppercase and lowercase versions of words as the same word. For example: “Man” and “man” should both be counted as “man”.

Example output:

the, 1234

thou, 453

etc.

Instructions:

* Write map reduce code to compute the values.  We recommend one of the following:
  + Download “MapReduce.java” from LMS.  This contains a poor-man’s MapReduce.  You provide java implementations of the Mapper and Reducer interfaces, while the provided code manages running a specified number of Mappers and Reducers in separate threads.  This is neither distributed nor fault-tolerant, but the approach is the same as more sophisticated implementations.
  + Or you may write a MapReduce to run on Amazon Web Services.
  + Or you may use Apache’s Hadoop, which provides a distributed fault-tolerant MapReduce you can deploy on your team’s machines or a single machine.
* Run your code with at least 4 different degrees of parallelism (eg 1 mapper vs. 4 mappers vs. 10 mappers vs. 100 mappers etc).

Submit:

* Team’s code.
* In your report:
  + A short description of your running environment (“Python AWS” or “provided MapReduce on dual core 3.2 GHz running windows w/1GB RAM”, etc)
  + The amount of time it took to run the mapreduce job with different degrees of parallelism.
  + The answers the following questions.

1. What is used as the keys in the mapper tasks’ maps?
2. Based on your keys, how many concurrent reducer tasks were generated by the framework?
3. What is the least number of concurrent map tasks you could create to solve this problem and why?
4. What is the largest number of concurrent map tasks that you could create to solve this problem and why?
5. If hardware was not limiting, what is the largest number of map tasks that could be run in parallel for this problem and why?
6. For your code, what number of map tasks provided the fastest completion time for your MapReduce application?
7. What 10 words appear the most frequently in the complete works of Shakespeare and how often does each word appear?

**Problem #2 (MapReduce)**

Look at the ~500,000 food reviews in food.txt.zip.  For this problem, you will find the average review rating for all “coffee” products.

Instructions:

1. Download foods.txt.zip (~122MB) from eLearn (source: <https://snap.stanford.edu/data/web-FineFoods.html>)
2. Create a MapReduce class that will return:
   * 1. The total number of reviews containing the word “coffee”.
     2. The average review rating for all products mentioning “coffee”.

Submit in your report:

1. The total number of reviews containing the word “coffee”.
2. The average review rating for “coffee” reviews.

Also submit:

1. Your team’s code.