**ECE 3822: Engineering Computation II**

# Homework No. 15: Map/Reduce Applications

As an introduction to this assignment, please review the following tutorial:

*http://stackoverflow.com/questions/12375761/good-mapreduce-examples*

Implement solutions to the following problems using the map and reduce functions in Python:

1. Count the number of times each word occurs in the EEG reports database. You might find this tutorial useful:

*http://programminghistorian.org/lessons/counting-frequencies*

Compare the efficiency and complexity of your code to the imperative style of coding we used previously to solve this problem. Store this as comments at the top of your source code file.

1. Implement the “find common friends in Facebook” example discussed in the stackoverflow tutorial. However, apply this code to the EEG reports database to find the two documents in the database with the most words in common.

To test your code, run this data set first: */data/courses/ece\_3822/2019\_spring/eeg\_reports*. This set contains about 23,000 files and should run in a few minutes. Email the course instructor your results to verify that your program works. Examine the output – it will be clear if the closest match you found makes sense.

Submit your solution in: */data/courses/ece\_3822/current/homework/hw\_15/<lastname\_firstname>*. Submit your solutions as p01.py an p02.py respectively.

Once you think you have a good, efficient solution for no. 2, for an addition 1,000 points on your homework grade, you can opt to have your code tested on a one million document database. Your code must find the two closest documents in less than one hour of CPU time on nedc\_000.

However, if your code fails this test, you will lose 200 points on your homework assignments. Consider this the price you pay for wasting our time ;) It can fail for a variety of reasons, such as using too much compute time or running out of memory.

The reason behind this latter constraint is that I want you to be able to accurately project the run-time of your algorithm on a large set before you actually run that data. This will require some careful thought and code analysis. Try plotting the speed of your code for N = 10, 100, 1000, and 10,000 documents and then projecting what it might take to run 1,000,000 documents.