# New York University
# CS-GY 6513-D: BIG DATA
# Spring 2024
# Prof. Amit Patel

## HW#2 (5% weight towards final grade)

**Submission Details:**

**This is a group assignment. All team members are expected to participate effectively towards solutions of assignment questions. Your team members are going to be the same as your project teammates. We encourage you to discuss freely with each other and form solutions.**

There are 2 questions in this assignment, both of which require you to use map reduce to solve them. You must submit a zip file which contains 2 folders, one for each question (name these folders with the question number). Each folder must contain the mapper, reducer and the output file which you have created to solve the problem, along with a document with screenshots of **your** hadoop environment which shows you've ran the code on your system.
Any additional information which explains the logic of your code may also be added to this document and this will be taken into consideration if your code doesn't work, although it is optional. The final zip file must be named as APC_HW2.zip (replace APC with your group members first name first characters. In this example, APC is the team of Andy, Peter, and Charlie).

Please note that all students of the group need to submit the assignment on Brightspace.

## 1. Stratified Sampling (50 points)

For this question, you will use sampling.txt. The file contains a bunch of text from a prose. Using stratified random sampling, you must collect 200 samples from this text. Below are a couple of links for understanding stratified random sampling better:

https://www.scribbr.com/methodology/stratified-sampling/
https://www.investopedia.com/terms/stratified_random_sampling.asp

The main idea is to divide your population into groups or strata and then randomly select an equal number of samples from each group.
The samples which you must collect are going to be lines from the text file. You are going to be dividing the text into 3 strata:

i) Lines with even number of words (what you consider as a word is up to you, for example you may ignore all other special characters in your count and

only count proper words with characters, or you may choose to include them)

ii) Lines with odd number of words

iii) Lines with one or two words (Consider this as a separate group and not part of the odd or even group)

Then, randomly choose your samples according to the algorithm. How you want to account for that randomness is up to you (Make sure you form a way where if you run the code multiple times you would not end up picking the same order of lines).
The output file should contain all 200 random samples. Using map reduce, perform this task and attach an additional text/word file explaining your logic and assumptions, or explain it well in the code comments.

## 2. One Way Relationships (50 points)

For this question, you will have to use relations.txt. This file contains alphabets with relations to each other, denoted with the "->" icon.

For example,

A->B

means A is related to B and,

A->B->C

indicates A is related to B and C, and B is related to C (basically an alphabet is related to all the alphabets to the right of it).

The objective of this map reduce exercise is to find all one-way relationships within the file. If A->B exists and B->A does not exist, then A and B have a one-way relationship as only A is related to B but B is not related to A.

For example, if the question is,

A->B->C

B->A

A->C->B

then the output should be, "A and C are in a one-way relationship"

**Things to note for both questions:**

1. No changes should be made to the input file.
2. You may not use any data structures which allow you to store values (lists, dictionaries, heaps etc). You may create an initial version which uses them to try and test if you want. This is to make the solution use as little memory as possible.

3. Take care of duplicates for Q2 like in the above example.
4. Focus on two things to get the optimal solution for Q2:
    a. Find a nice format to arrange your data in your mapper before passing it to the reducer.
    b. Find a way to make this into a counting problem.