The overall project for this semester is to simulate a search engine over a collection (“corpus”) of documents. This project will be divided into three phases. The requirements described here are for Phase 1.

Phase 1 has a five main tasks and one auxiliary task but these will be used in subsequent phases as well:

**Task#1)** You will need to maintain your own corpus of documents for the semester. To do so, come up with 10 neutral (ie no controversy) queries (for example: Who was the 16th President?) that you will submit to a search engine of your choice. You are to then download the first 20 (non-controversial) webpage responses that the search engine returns with, for each of the 10 queries (this is manually done; you have to do it one by one). There will be a total of 200 html files. (We will be discussing shortly in class how to process these using the Java Regex package. You may NOT use 3rd party code. You MUST write your own. You do not need regex necessarily but it does provide much more concise code.)

**Task#2)** Identify a Stoplist (either download or compute in a separate code on your own) and store it in a hash structure. (As mentioned later, you will need program code to output your hash structure to a output text file.

**Task#3)** Compute an Inverted Index collectively storing info for All of the above files. See the following links for an explanation of what an Inverted Index is (and is not, ie forward index):

https://www.geeksforgeeks.org/inverted-index/

https://www.geeksforgeeks.org/difference-inverted-index-forward-index/

You are to use either hashmaps or hashtables (separate email will provide tutorial links) for storing the inverted index of your corpus. What information should you store in the inverted index for each significant (ie non-stopword) found in one of your documents? a) the word; b) the name of document found in; c) a vector specifying for each occurrence of the word in a document, how many words from beginning of document was it found (for this count include even the stopwords). You need to do this for every word in every document that is not a stopword.

**Task#4)** The code for each phase has to be compiled using javac (jdk compiler) and executed using the java command (jdk runtime environment). Important names of files etc. will be provided on the command line of the java command using “flags.” Details about the usage of flags for this phase will be discussed below. Further Code issues will be explained in a separate email on Command Line Parsing. Please note that ALL phases of this project will be run from the command line only.

**Task#5)** You will need to demonstrate the ability to “query” your inverted index for such information as a) does a specific word appear in any document? b) how many documents (and which) does a given word appear in; c) how many times (frequency) does a word appear in a given document. For example,

**SEARCH WORD** -- would search the Inverted Index for the given WORD and return with which documents does the word appear in and specifically how many times appears in that document.

**SEARCH DOC** -- would search Inverted Index for the given Document and return all words found in that DOC with specifically how many times appears in that document.

**NOTE:** For the above commands, you will need to pass other parameters via the command line using appropriately named flags.

**Task#6)** The system should be able to printout the inverted index pertaining to a given document.

**The submission process:**

Phase 1 of your project sets up the necessary data structures for future experimentation. Your code must compile on the CommandLine by using oracle jdk, javac compiler using the command: javac \*.java.  This implies that ALL of your java files are located in the same place. So, when submitting, DO NOT submit files in different directories (which code editors tend to do). Likewise COMMENT out the "package" command from the first line of each .java file submitted. (Many editors insert such a package command. You must comment out because CommandLine compilation tends to ignore it and this could cause errors otherwise.)

For consistency across your projects, call the primary (main) class of your project as SearchEngine and the name of its file, SearchEngine.java This file will remain so named for Phase 2 and 3 as well. The CommandLine to run your program will be: (note: CommandLine flags can be listed in any order; your code should be able to handle that)

java **SearchEngine** -CorpusDir **PathOfDir** -InvertedIndex **NameOfIIndexFile** -StopList **NameOfStopListFile** -Queries **QueryFile** -Results **ResultsFile**

where **PathOfDir** is the user’s choice of where the Corpus is installed and where the InvertedIndex can be outputted to; **NameOfIIndexFile** is the name of a text file where the InvertedIndex can be outputted to; **NameOfStopListFile** is the name of a text file containing the stoplist; **QueryFile** will contain the queries for the inverted index and the outputted results to these queries will be written to the **ResultsFile**.

How to submit these files will be further discussed next. Since the files may be large, for this phase ONLY, you can email from your nonCuny email (that you have told me about) to the Class Gmail. Because the outputs of some of the steps are large, it will be required to email the project in four emails, now described.

**1)** You should have already downloaded all the necessary html (data) files for your corpuss stored in directory **PathOfDir**. Again, you were to choose 10 neutral (non-controversial queries) and for each of these, download the first 20 noncontroversial (webpage) results returned by the search engine that are in *html* format. (This was explained weeks ago.) Anyway, now also prepare a Word document listing the actual 10 queries and then attach to one email this word document and a compressed file containing all the html results from these queries that you downloaded. The Subject Line of this first email should be:

**Subject: IR Phase1 Corpus <PHASE1 DUE-DATE>**

2) In a Second Email, you will email the stoplist you are using, **NameOfStopListFile**, and also attach a compressed version of the inverted index that your program should output to a text file **NameOfIIndexFile**. (You can manually compress it after your project outputted it.) Your system will print out the inverted index into a text file (described soon). You will compress that file and attach as well to the second email. The subject line for this email should be as follows:

**Subject: IR Phase1 StopList And Index <PHASE1 DUE-DATE>**

**3)** Then, In a Third Email for the actual project code submission, you should email all .java files (properly commented) attached to one Qmail to Class Gmail. You should also include a Readme.txt (and/or UserManual) file which explains how to compile your code from the CommandLine using the javac compiler (again all java files should be assumed to be in the same directory) and run your code using the java runtime command (explained above at beginning of this document.) The subject line for this email should be as follows:

**Subject: IR Phase1 Code <PHASE1 DUE-DATE>**

4) A fourth email involves searches on the *inverted index* (as described in the original email requirements documents.) On the CommandLine, you will specify a text file **QueryFile** that holds queries that will be conducted on the Inverted Index. The information of all those queries will as well be outputted to a text file **ResultsFile**. There will be two types of queries: (the keywords to submit queries will be standardized here)

a) Does a specific word appear in any document? The output would be which documents do they appear in. This query would appear in **QueryFile** simply as:

**Query <Term>** where <Term> without the “< >” is the word you are seeking. You should search the inverted index of your Corpus and output the query itself (i.e. Query <Term>) following by the results to text file **ResultsFile**.

b) How many times (frequency) does a word appear in a given document? This query would appear in **QueryFile** simply as:

**Frequency <Term>** where <Term> is a in a) and output the query itself following by the output to text file **ResultsFile**. Here too you should use the information provided by the InvertedIndex.

The subject line for this email (with **QueryFile** and **ResultsFile** attached) should be as follows:

**Subject: IR Phase1 Queries and Results <PHASE1 DUE-DATE>**