CMSC 476/676
Homework 3 - work individually
Due: Timestamp on or before 11:59pm Wednesay, April 11, 2019

The objective of this assignment is to build an index for the documents in your collection. In this sense, an index is also called an inverted file: instead of looking up a file to see what terms occur in it, we're looking up a term to see what files that term occurs in.

To execute your program, it should be sufficient to type:

```
index input-directory output-directory
```

where the input-directory contains the original files (unprocessed) and the output-directory contains all generated output files (temporary files created along the way should be deleted to save space). The index command may be written in the language(s) of your choice.

If you're using older phases of the project to do this phase, you can leave out the parts that produce output this phase doesn't require.

Terms and Term Weights You should use the terms, and term weights, generated by Homework 2. If you are not happy with your Homework 2, you may use any classmate's Homework 2 as your starting point. Mention this in your report.

Output Files The goal is to build a pair of files of **fixed length records**. These files should be in ASCII for easier debugging. The dictionary records may be in either hash_order or alphabetical order. The dictionary should contain three fields:

the word
the number of documents that contain that word (this corresponds to the number of records that word gets in the postings file)
the location of the first record for that word in the postings file

If you want to make some asumption about the maximum length of a word, mention it in your report.

The second out put file, the postings file, should contain two fields:

the document id
the normalized weight of the word in the document

Algorithm You may implement either a memory based or sort based or multi-way merge algorithm. Measure the time the index function takes on a varying number of documents (10, 20, 40, 80, etc.) and plot a graph of the indexing time as a function of the number of documents in your report. The timings should be done on the FULL indexing process (starting with the raw documents). Time should include elapsed time as well as CPU time, as given by the UNIX time command, for example.

Example If we have a small corpus with three terms, cat, dog and rat, and three documents numbered 1,2,3, the term document matrix (TDM) would look like the following. I've put in some example weights.

	1	2	3
cat	0.2	0	0.3
dog	0	0.1	0.1
rat	0.1	0	0.4

The dictionary file then looks like:

```
cat
2
1
dog
2
3
rat
2
5
```

Where, taking dog as an example term, there are two entries in the postings file for that term, and they begin at position 3 in the postings file.

The postings file looks like:

```
1,0.2
3,0.3
2,0.1
3,0.1
1,0.1
3,0.4
```

Note that there is a single line (record) in the postings file for each non-zero entry in the TDM.

Program Testing
Use the same set of files as in Phases 1 and 2.

Program Documentation.
After your internal documentation (comments) are complete, write a report which provides an approximately 4 page summary of your program. In particular, discuss how you extended/improved the preprocessing. Discuss the efficiency of your approach in terms of your timings on a linux machine (use the time command) as a function of number of documents indexed. Also, discuss how the total size of the output files, dictionary+postings, compares to the size of the input files, for various numbers of input files. The entire document should be 4 pages maximum. We will primarily be grading from the report, so make sure it clearly describes what you did and what your program's output and efficiency.

Hand In
Include your code (including shell scripts), the report, and the first and last 200 lines of the dictionary and postings files. Please name the zip file in the form LastnameFirstnameHW3.zip . Email the zip file to the TA

Late Policy
10% deduction per 24 hours.