Tomer Levy & Neil Malu (tl455 & nmm182)

Assignment 2: Indexer

The way our indexer works is that it first starts off by recursively going through every file in the given directory. For every file, and every token inside of this file ;the input will be stored into a master hash table. This hash table contains 36 slots; one for each letter and one for every number. When it is being inserted, the word that is found within a file will also be set to lower case; if an existing word is trying to be added to the hash table, rather than creating a new node for it, the existing node will simply increment by 1. Next, a second hash table is made to keep track the number of repeats. The program will stick nodes into this hash table with index #'s that correspond to the # of repeats. When everything is sorted and properly chained, it will take all this information and stick it into the output file.

Efficiency:

0(#characters in file) – gathering tokens in the file takes 0(characters) time (tokenize).

0(n^2)- to put tokens into a hash table correctly, IE alphabetically the run time needs to be O(n^2) where n is #tokens.

Total Eff: O(n^2)

Space:

To collect tokens takes O(n) space, where n is # tokens.

Collecting tokens for our second hash table will also take O(n) space. Scattering nodes and sticking them into our second table takes O(n) space, where n is number of tokens in a file.

To Run, in the terminal; input make into shell, then ./index <file_name> <directory>