Our indexer program uses code from our tokenizer and sorted-list programs to index files. Tokenizer goes through each file we parse and returns a normalized string of tokens (normalized as in each token is separated by one white space). This takes linear time to scan through each file once.

Our sorted-list implementation takes linear time to insert each word and its respective file counter. We did not have time to implement an AVL tree which would have been a lot more efficient for inserting words. We used a sorted-list of sorted-lists. Each word node would have a sorted-list of files and the frequency that the word would appear in the file. We sorted by alphabetical order of the filenames.

Upon output to the inverted index, we had to parse through each word node and then their respective sorted-list of files. Since we did not sort by frequency before, we had to do it here, by keeping track of the highest count at all times. This takes n^2 time unfortunately.