1. Our Name and UNI
   Andrew Shu ans2120@columbia.edu
   Ran Bi rb2651@columbia.edu
2. Files that we submit
   a. Jave code files
   b. Makefile
   c. Readme
3. How to run our program
   a. Unpack the .tar.gz archive.
   b. Build the program using 'make'
   c. Run the program using 'make exec'
   d. (All output goes into the 'bin' directory)
4. Internal design of our project
   a. First, we read the documents in the dataset and get every word occurring in these files. We use TreeMap to store the words and corresponding occurrences. Then generate COMMON&WORDS files according to their occurrences. Meantime, we build six tables using HashMap: 1. docIds: store the name of every document and its unique ID; 2. wordIds: store every word and its unique ID; 3. idWords: the reverse table of wordIds; 4. wordDocs: store every word ID and the IDs of documents which contain this word; 5. docWords: store a doc ID and all the word IDs that it contains. 6. multiwordDocs: holds pairs of word IDs that show up in at least 1 document together
   b. We create a data structure called Itemset that is just a bitmap. Each word corresponds to 1 specific bit position in the bitmap. However since there can be many words, we store a "disjoint bitmap" so instead of storing the entire bitstring where len(bitstring) = # words, we store a list of tuples (rangeId, bitmap) where rangeId is the chunk of bitmap that the tuple represents. Each range is 32 bits since Java has 32-bit integers. Itemset is a useful data structure because it lets us do fast bit manipulations like bitwise AND and OR.
   c. Do Apriori algorithm to find LargeItemSets. First, we get the values of minsup and minconf from the user. Generate the one item largeItemset from the file of sortedwords, which is also the WORDS file. Then, we use the function of Apriorigen to generate the candidate itemsets and prune some sets according to minsup.
   d. Generate association rules according the LargeItemSets. Due to we use the class of Itemset to store every largeItemset, we get the support value for every Itemset. Then for every Itemset, we compute the support value for every word in this Itemset. And we can get the confidence for every association rule by divide the support of Itemset by the support of each word. After that, prune some association rules by minconf.
   e. We use a smaller index for 20newsgroups since our larger faster index crashes it. However the larger index is very useful. We are limited by Java's heap allocator (we tried using "java –Xmx3072m" for instance, but it was unable to allocate that heap, even using virtual memory swap space.