1. Our Name and UNI
   Andrew Shu ans2120@columbia.edu
   Ran Bi rb2651@columbia.edu
2. Files that we submit
   a. Java code files
   b. Makefile
   c. Readme
   d. run.sh
3. How to run our program
   a. Unpack the .tar.gz archive.
   b. Build the program using 'make'
   c. Run the test program using 'make test'. OR use:
      run.sh <website> <tes> <tec>
      e.g.,
      run.sh diabetes.org 0.6 100
4. Internal design of our project
   Part1:
   a. First, we build two hash tables: category hierarchy table and query table. In these tables, we store the hierarchy of categories and queries corresponding to each category.
   b. For every database, we compute its coverage values and specificity values for "Computers", "Health", "Sports". This is where we perform the queries. After these, compare these values with specified Tec and Tes. If both of coverage and specificity values exceed the specified values, then add this category to the result and go on to next loop.
   c. In the next loop, we compute the values for subcategories corresponding to category whose values are larger than specified values. Then if there are categories whose values are larger than specified values, add these subcategories to the result.
   d. If there is no category satisfying above description, the result is "Root".

   Part2:
   a. Part 2 is integrated into Part 1. So during the query part (Part 1-b), we store the top-4 hits for each query, for every node in the classification hierarchy. This corresponds to the "Document Sampling" of Part 2a in the assignment.
   b. Next we create a "Content Summary" using Lynx to retrieve the term document frequencies for all queries associated with the nodes that we visited (i.e., nodes that appear along the paths to the final categorizations of the database). To get these terms, we make a set of results (eliminating duplicate Results by comparing URLs), and for every category that appears in the Part 1 categorization, we extract the desired category names using StringTokenizer. Then use Lynx to retrieve the URLs and parse HTML for word count.
   c. Finally, write this to a file.
5. Yahoo! BOSS Application ID
   SeJQZ5fV34F7ohb4ONiSH9bbdWH9RtbodjvH_cN_BRj9QWEgfSFLW1h.Jkj0i52LT6I-