**Project part 1**

**Due October 24, 2017**

Write a program that reads a document and outputs all the index terms that occur in it in lexicographic order.

The program should have one command parameter <file name>.txt (for the input file)

The document is a simple HTML file.

1. Extract only terms between tags. Tags should be ignored. There will be no fixed pattern of tags, so make sure your code can handle all tags.

2. All index terms will be lower case.

3. Do not index the following stop words: and, a, the, an, by, from, for, hence, of, the, with, in,

within, who, when, where, why, how, whom, have, had, has, not, for, but, do, does, done.

4. Do not index a single character followed by space.

5. Use hyphen as an end of token. So the word data-base will be stored as two terms data and base. Unfortunately the date 1970-80 will become two numbers 1970 and 80.

6. Get rid of single and double quotes (“, ‘) or parentheses (( ,  [) when they are the first characters of a token

7. Get rid of single and double quotes (“, ‘) or parentheses (( ,  [) when they are the last characters of the token.

8. A comma, a period, question mark, colon, semicolon, or an exclamation mark followed by space should not be included in a token (leave them in the token if they are not followed by a space)

9. Do index numbers (3,1111 will be stored as a single index term)

10. Delete all apostrophes in a token. So car’s, and cars’, will be changed to cars and then stemmed to car.

11. Do the following minimal stemming that deals with plurals and third person:

a. **if** word ends in

“ies” but not “eies” or “aies” then “ies”->“y”; (skies becomes sky, dies become dy)

b. **else** in “es” but not “aes”, “ees” or “oes” then “es”->e; (retrieves becomes retrieve)

c. **else** in “s” but not “us” or “ss”   then “s”->NULL (doors becomes door, success stay success)

**endif**

12. Your program should be able to handle any extra space or blank lines in the document.

In the next part of the project you will create an index for a collection of movie reviews. I uploaded to Blackboard some of these files.

**Submission Directions**

You may write the code using C, C++, Java, or Python. Your program should compile on remote.cs.binghamton.edu. No exceptions. If you have trouble accessing the remote server please contact CS system administrator.

Your need to submit a .tar.gz file (not only .tar) which should follow the following naming convention. <usrid>\_<language>\_part1.tar.gz, after unpacking this .tar.gz it should have a directory named <usrid>\_<language>\_part1. Example: if your user id is psaha4 and you are writing the assignment in java then your submission file name will be psaha4\_java\_part1.tar.gz. For c++ and c developers it will be psaha4\_cpp\_part1.tar.gz and psaha4\_c\_part1.tar.gz respectively.

Your submission folder should not contain input files, object or class files and readme file. Presence of these file will cost you 5 points. It should contain a make file and source code files.

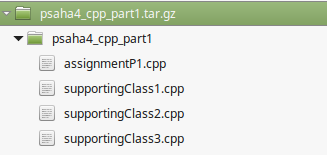
For Java developers your entry point class, which contains the main method, should have the name as “asignmentP1.java”. So make sure your makefile will produce a assignmentP1.class file after compilation and it can be run by the following command java assignmentP1 <file path>/<input file name> <output file name>

All your .java files should be inside a single folder, don’t create many packages to place your java files. All source code files should be inside a single directory like below.



For C++ and C developers please write your makefile such a way that it produces object file with name assignmentP1.obj and it can be run with the below command ./assignmentP1.out <file path>/<input file name> <output file name>

All source code files should be inside a single directory like below.



Hard coding of any filename and file path will cost you 5 points, so make sure your program can take parameters dynamically.

Your program should generate the file dictionary.txt to an input file mentioned in the command line arguments.

Please ask questions regarding assignment and submission instruction to TA, don’t assume anything.

**Plagiarism Policy**

All your code will be checked for similarity to other submissions using Moss. Programmers have an uncanny knack of reproducing the same code that they have seen before. So you are advised not to look at each other's code.