COMP9319 2017s1 Assignment 3: Indexing and Searching Multiple Files

Your task in this assignment is to create a search program that indexes and searches multiple ASCII files in a case insensitive manner. For simplicity, you may assume these files are English documents. The advanced part of this assignment is to perform concept search on the given files.

Your C/C++ program, called **a3search**, accepts the path to a destination folder that contains the target files; the path to a folder of index files; an optional argument `-c cnum` specifies the search is a concept search where cnum is a decimal number between 0 and 1 (and can be equal to 0 or 1); and **one** to **five** quoted query strings (i.e., search terms) as commandline input arguments. A search term may include English alphabets (i.e., you do not need to consider spaces, digits and symbols). Each search term is at least 3 characters and can be up to 256 characters. Using the given query strings, it will search on all the target files, and output (to standard output) all the files (the filenames only) that contain ALL input query strings (i.e., a boolean AND search), ranked on how frequently the search term has appeared in the files in descending order.

For multiple search terms, it will rank the results based on the average frequency per search term for each file. If their frequencies are the same, the filenames will be output in ascending order (lexicographically). When concept search is involved, we assume the frequency of each concept match is cnum. Hence, when cnum is equal to 1, its ranking will be treated exactly the same as the one of a keyword match. When cnum is equal to 0, the contribution of the frequency counts from the concept matches will be disregarded during ranking (i.e., they will be ranked below those keyword matches).

The total size of all the files (maximum 2000 files) inside a destination folder will be max 200MB. The total size of all your index files inside your index folder cannot be more than 1MB or the same size of all the given files (whichever is larger). If your index file is larger than this maximum, you will receive zero points for the tests that using that folder of given files. You may assume that the index folder and its files inside will not be deleted during all the tests for a given destination folder, and the name of the index folder will not be changed for that given destination folder. Therefore, to save time, you only need to generate the index file when it does not exist yet.

Your search may ignore stop words and shall include the stemming process. External library may be used for this process (and if you do, please submit it as part of your solution). You shall not perform just mechanical substring search such as Boyer Moore over the original text as this will result in lots of meaningless matches.

To make the assignment easier, we assume that there is no subfolder inside the destination folder. Your output shall be one line (ending with a '\n') for each filename. No filename will be output more than once.

We will use the `make` command below to compile your solution. Please provide a makefile and ensure that the code you submit can be compiled. Solutions that have compilation errors will receive zero points for the entire assignment.

```
make
```

Your solution will be compiled and run on a typical CSE Linux machine e.g. wagner. Your solution should **not** write out any external files other than the index folder and its index files inside the folder. Any

solution that writes out external files other than those inside the index folder will receive zero points for the entire assignment.

During the auto marking, fewer than one fifth of the tests will involve concept search. Concept search marking will then be double checked manually (to allow a small degree of flexibility for marking). To tackle concept search, you may use external C/C++ software library (for example, Wordnet) in your solution. If you use any external library, please submit it as part of your give submission, and the above make command shall make the complete solution including the compilation of the external library. I have increased the maximum quota of give submission to 50MB per student. If you believe that your solution is larger than this limit, please contact your lecturer in advance.

Example

Suppose there are five files in `~MyAccount/XYZ/`:

```
file1.txt: Apple investors brace for first decline in iPhone sales
file2.txt: Earnings deluge threatens to overwhelm investors
file3.txt: Apple Watch from Apple disappoints some investors
file4.txt: People are buying iPhone
file5.txt: Invest in Apple shares because of their apps?
file6.txt: Go Apple! Apple! Apple! Go iPhone!
```

First example:

```
%wagner> a3search ~MyAccount/XYZ dummy.idx "apple"
file6.txt
file3.txt
file1.txt
file5.txt
%wagner>
```

Another example:

```
%wagner> a3search ~MyAccount/XYZ dummy.idx "INvestor"
file1.txt
file2.txt
file3.txt
%wagner>
```

Yet another example:

```
%wagner> a3search ~MyAccount/XYZ dummy.idx "apple" "investor"
file3.txt
file1.txt
%wagner>
%wagner> a3search ~MyAccount/XYZ dummy.idx "apple" "APP"
file5.txt
%wagner> a3search ~MyAccount/XYZ dummy.idx "apple" "APP" "sales"
%wagner>
```

And the last example:

```
%wagner> a3search ~MyAccount/XYZ dummy.idx -c 0.5 "iPhone"
file6.txt
file1.txt
file3.txt
file4.txt
file5.txt
```

```
%wagner>
%wagner> a3search ~MyAccount/XYZ dummy.idx -c 0.2 "invest" "iphone"
file1.txt
file5.txt
file3.txt
%wagner>
```

Performance

Your solution will be marked based on space and runtime performance.

Runtime memory is assumed to be always less than 32MB. Runtime memory consumption will be measured by `valgrind massif` with the option `--pages-as-heap=yes`, i.e., all the memory used by your program will be measured. Any solution that violates this memory requirement will receive zero points for that query test. Any solution that runs for more than **300 seconds** on a machine with similar specification as *wagner* for the first query on a given destination folder will be killed, and will receive zero points for the queries for that folder. After that any solution that runs for more than **5 seconds** for any one of the subsequent queries on that folder will be killed, and will receive zero points for that query test. We will use the `time` command and count both the user and system time as runtime measurement.

Documentation

You will be marked on your descriptions in README.txt of the design of your search, index files and how your solution works (including if and what external libraries that you have used). For example, a good design should separate the code of searching, joining and ranking so that they can be easily extended. Your source code will be also inspected and marked based on readability and ease of understanding.

Assumptions/clarifications/hints

1. Please open the files in the given destination folder path as read-only in case you do not have the write permission to some of these files.
2. Marks will be deducted for the output of any extra text, other than the required, correct answers (in the right order). This extra information includes (but not limited to) debugging messages, line numbers and so on.
3. You are allowed to use one external index folder containing all index files to enhance the performance of your solution. However, if you believe that your solution is fast enough without using any index files, you do not have to generate the folder. However, even in such case, your solution should still accept a path to index folder as one of the input argument as specified.
4. A filename will not be unreasonably long. You may assume that it is up to 256 characters.
5. As shown in one of the examples above, output just one newline character if there are no matches for the query.

Marking

This assignment is worth 100 points. Below is an indicative marking scheme:

Component	Points
Auto marking	95
Documentation	5

Bonus

Bonus marks (up to 10 points) will be awarded for the solution that achieves 100 points and runs the fastest overall (i.e., the shortest total time to finish **all** the tests excluding the first query for each destination folder - it will not count the indexing time). This solution will be shared with the class. Note: regardless of the bonus marks you receive in this assignment, the maximum final mark for the subject is capped at 100.

Submission

Deadline: Thursday 25th May 23:59. Late submission will attract 10% penalty for the first day and 30% penalty for each subsequent day. Use the give command below to submit the assignment:

```
give cs9319 a3 makefile *.h *.c *.cpp *.tar README.txt
```

where *.tar above assumes that you use an external library that is packaged as a tar file. Your make command shall untar the file and compile the library and hence the final solution accordingly. Please use "classrun" to check your submission to make sure that you have submitted all the necessary files.

Plagiarism

The work you submit must be your own work. Submission of work partially or completely derived from any other person or jointly written with any other person is not permitted. The penalties for such an offence may include negative marks, automatic failure of the course and possibly other academic discipline. Assignment submissions will be examined both automatically and manually for such submissions.

Relevant scholarship authorities will be informed if students holding scholarships are involved in an incident of plagiarism or other misconduct.

Do not provide or show your assignment work to any other person - apart from the teaching staff of this subject. If you knowingly provide or show your assignment work to another person for any reason, and work derived from it is submitted you may be penalized, even if the work was submitted without your knowledge or consent. This may apply even if your work is submitted by a third party unknown to you.