COMP 345 Advanced Program Design with C++

Fall 2017 (Section N)

Assignment #1

Due date (Moodle Submission): October 3
Due date (Lab Demo): October 4 & 11

This is the first of four programming assignments. Note that the assignments are not 'stand-alone', but rather build on top of each other to form a bigger project. Hence, subsequent assignments will require you to extend and/or refactor code you have written for the previous assignments.

The assignments will focus on a number of tasks that are typical for large-scale data analysis: For example, you will implement (simplified) versions of a search engine, a document analysis and a recommendation system.

Task 1: "Document Indexing" (5%). Your first task is to create a so-called *document-term matrix* from a number of input documents. You have to build a *dictionary*, which is a set of all words that appear across all documents. To build the dictionary: *(i)* process all input documents; *(ii)* split the input text into tokens using whitespace and punctuation; and *(iii)* convert all characters to lower case. Count the occurrence of each token per document and print the results in a table sorted by token, like this:

Dictionary	**Doc$_1$**	**Doc$_2$**	**Doc$_3$**	...
adventure	1	0	5	·
c++	1	0	1	·
dummies	0	3	2	·
in	3	6	1	·
java	0	1	3	·
programming	4	4	0	·
...	...	...	...	·
Total	83	97	111	·

Input. Your program has to first read a configuration file (e.g., `index.txt`) that specifies which documents to index. Each line in this file contains the filename of a document. That is, you have open each document listed in this configuration file and add it to your document-term matrix.

Processing. You have to compute two versions of the document-term matrix: *(i)* A complete matrix, which contains all tokens; and *(ii)* A filtered matrix, where you remove all tokens that come from a list of so-called *stopwords* (this list will be available on Moodle).

Output. Print the two versions of the document-term matrix as defined above to `cout`. Use stream manipulators to format the table entries: dictionary words left-adjusted, fixed length, numbers right-adjusted, same width for each column. Additionally, print a legend indicating the file name for each document (column in the table).

Coding guidelines. Submit a single file `indexer.cpp`, which must include your indexing functions, as well as a `main()` function to run your code. Include the group members (names, ids) in a comment at the top of the file. You are free in the choice of an IDE, but your code must be standard, cross-platform C++ code; i.e., it must be possible to compile your code on any platform using the `gcc` compiler. **Do not** submit any IDE-specific project files (such as Eclipse `.project` files).

Submission. You must submit your code electronically on Moodle by the due date (late submission will incur a penalty, see Moodle for details). You must also demo your code to the marker in one of the lab sessions (time slots for the demo will be reserved through Moodle). Note that all group members must be present for the demo and ready to answer questions about the code.