CSCI 4963/6963 — Large-Scale Programming and Testing
Homework 3 (document version 1.3)
Inverted Indexes and Snippet Generation in C

Overview

- This homework is due by 11:59:59 PM on Tuesday, December 12, 2017.
- This homework will count as 10% of your final course grade.
- This homework is to be completed **individually**. Do not share your code with anyone else.
- You **must** use C for this homework assignment, and your code **must** successfully compile via `gcc` with absolutely no warning messages when the `-Wall` (i.e., warn all) compiler option is used. We will also use `-Werror`, which will treat all warnings as critical errors.
- Your code **must** successfully compile and run on Submitty, which uses Ubuntu v16.04.3 LTS. Note that the `gcc` compiler is version 5.4.0 (`Ubuntu 5.4.0-6ubuntu1~16.04.4`).

Homework Specifications

In this third homework, you will make use of the inverted index files generated by the previous homework. More specifically, using C, you will write a program that reads in a directory of inverted index files, where each inverted index file follows the format specifications introduced in Homework 2. As you read each inverted index file, incorporate the data into an in-memory structure that you then use to support queries. Another directory contains the original documents, from which you will generate snippets to be displayed to the user.

Document Directory

To identify documents in the inverted index files, document IDs (i.e., `docIDs`) are mapped to filenames via a `contents.txt` file, which follows the format shown in the following example:

```
1 book-dorian-gray.txt
2 book-1984.txt
3 book-kalevala.txt
...
```

For snippet generation, each of the original indexed documents are stored in a directory called `documents`.

Inverted Index Directory

The inverted index files are stored in a directory called `indexes`. As noted above, you must first read each inverted index file from this directory, storing each index into a data structure of your choice (i.e., build your own and/or incorporate open source structures).

Each inverted index file is a regular text file with a `.index` filename extension. You can determine the set of inverted index files from the `contents.txt` file noted above, replacing the `.txt` filename extension with `.index`.

Handling User Queries

Given your in-memory inverted index, your program must provide a simple interface (described below) for accepting user queries. Your interface will return a summary of results that match the given query.

As with Homework 2, identify words in the given query as consisting of alpha characters and at most one internal single quote. Further, convert all query terms to lowercase. And to simplify this assignment, consider only the first five valid words detected, ignoring any search keywords beyond this.

In addition to identifying individual words in the search query, you must also identify bigrams (e.g., United States) and trigrams (e.g., Rensselaer Polytechnic Institute). Similar to Homework 2, bigrams and trigrams should not contain "stop words"; you may hardcode (or read from a file) the following 30 stop words:

```
the of to a and in said for that was
on he is with at by it from as be
were an have his but has are not who they
```

If you do read these in from a file, please be sure you submit this file along with your source code on Submitty.

Given that you will have at most five individual search keywords, you will have at most four bigrams and three trigrams as part of your search.

Query, Snippet Generation, and Output Requirements

Your program should provide an interactive command-line interface that asks users to enter search queries via a "`SEARCH:`" prompt. For each search query, show the number of matches (i.e., "hits") for each individual word, as well as each bigram and trigram (if any).
To give users an idea of the context within which their hits occur, generate snippets from the original document(s) as necessary. Only display one snippet per hit per document. For each document, generate at most three snippets; prioritize as follows to determine the top three snippets:

- Show trigram snippets based on trigrams in the given search from left to right.
- Show bigram snippets based on trigrams in the given search from left to right.
- Show individual word snippets based on trigrams in the given search from left to right.

A snippet must include the given trigram, bigram, or individual word, as well as the surrounding 10 words; more specifically, display the five words before and after the trigram, bigram, or individual word. Use the definition of word from above. Further, replace all newline `'\n'` characters with space characters, thereby displaying the snippet as one line. **(v1.2)** And deduplicate whitespace characters by replacing multiple adjacent occurrences of whitespace (i.e., identified by the `isspace()` function) with a single space character. Since the given inverted indexes do not have positional information, find each first occurrence of the trigram, bigram, or individual word in the given document.
(v1.3) To support auto-grading on Submitty, please have your program exit when the user enters "`QUIT`" (i.e., return `EXIT_SUCCESS`).
For your program output, match the output format shown in the example below:

```
Reading docID 1 book-dorian-gray.index...
Reading docID 2 book-1984.index...
Reading docID 3 book-kalevala.index...
SEARCH: winston
Number of "winston" occurrences is 455 [docID 2]
...the clocks were striking thirteen. Winston Smith, his chin nuzzled into...
SEARCH: winston smith
Number of "winston smith" occurrences is 6 [docID 2]
Number of "winston" occurrences is 455 [docID 2]
Number of "smith" occurrences is 26 [docID 2 3]
...the clocks were striking thirteen. Winston Smith, his chin nuzzled into his...
...the clocks were striking thirteen. Winston Smith, his chin nuzzled into...
...clocks were striking thirteen. Winston Smith, his chin nuzzled into his...
...win thy lovely maiden; Worthy smith is Ilmarinen, In this art...
SEARCH: Thanksgiving
Number of "Thanksgiving" occurrences is 0
SEARCH: Thanksgiving break
Number of "Thanksgiving break" occurrences is 0
Number of "Thanksgiving" occurrences is 0
Number of "break" occurrences is 38 [docId 1 2 3]
...I am not going to break my word to her. She...
...could occasionally nerve yourself to break. It was dangerous, but it...
...oat-loaf, causing him to break his knife, the only keepsake...
SEARCH:
```

Submission Instructions

To submit your assignment (and also perform final testing of your code), please use Submitty, the homework submission server. The specific URL is on the course website.

Note that this assignment will be available on Submitty a few days before the due date. Please do not ask on Piazza when Submitty will be available, as you should perform adequate testing on your own.

To make sure that your program does execute properly everywhere, including Submitty, read the following tip.

Output to standard output (`stdout`) is buffered. To ensure buffered output is properly flushed to a file for grading on Submitty, use `fflush()` after every set of `printf()` statements, as follows:

```
printf( ... );     /* print something out to stdout */
fflush( stdout );  /* make sure that the output is sent to a */
                   /*  redirected output file, if specified  */
```