Software Engineering 265
Software Development Methods
Summer 2022

Assignment 1

Due: Thursday, October 13, 11:55 pm by `git push`
(Late submissions **not** accepted)

Programming environment

For this assignment please ensure your work executes correctly on the Linux machines in ELW B238. You are welcome to use your own laptops and desktops for much of your programming; if you do this, give yourself a few days before the due date to iron out any bugs in the C program you have uploaded to a lab workstation. (Bugs in this kind of programming tend to be platform specific, and something that works perfectly at home may end up crashing on a different hardware configuration.)

Individual work

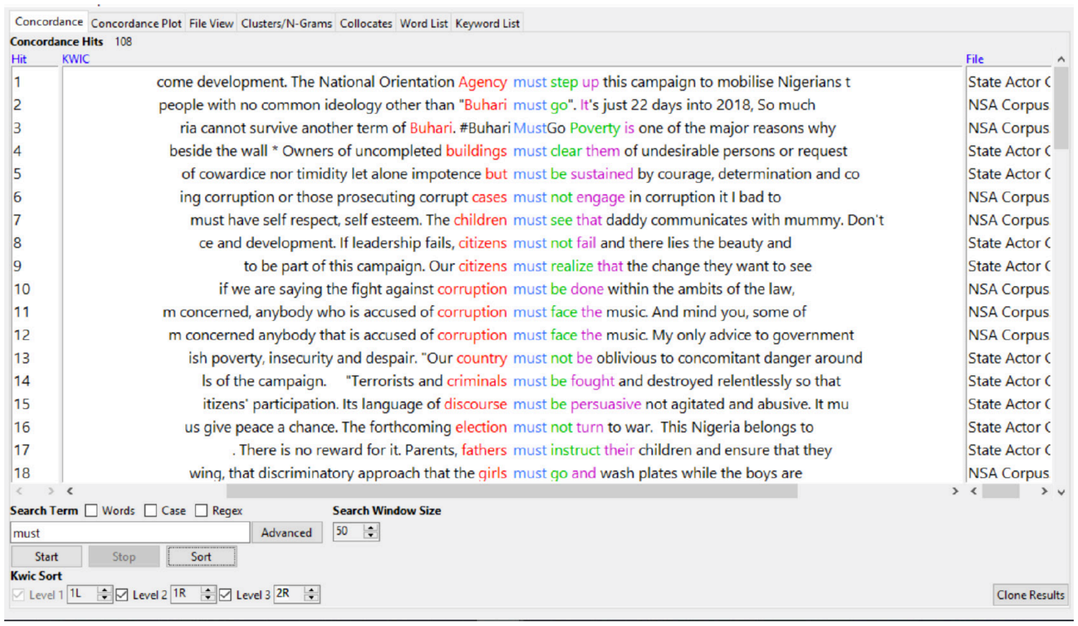
This assignment is to be completed by each individual student (i.e., no group work). Naturally you will want to discuss aspects of the problem with fellow students, and such discussion is encouraged. **However, sharing of code fragments is strictly forbidden without the express written permission of the course instructor (Zastre).** If you are still unsure regarding what is permitted or have other questions about what constitutes appropriate collaboration, please contact the course instructor as soon as possible. (Code-similarity analysis tools will be used to examine submitted work.) The URLs of significant code fragments you have found online and used in your solution must be cited in comments just before where such code has been used.

Objectives of this assignment

- Understand a problem description, along with the role played by sample input and output for providing such a description.
- Use the C programming language to write the first phase of a "concordance line" utility. The program will be named (rather unimaginatively) `concord1.c`.
- Use `git` to track changes in your source code and annotate the evolution of your solution with "messages" provided during commits.
- Test your code against the ten provided test cases.

This assignment: `concord1.c`

Throughout this term the assignments will focus on a particular kind of text indexing known as a concordance (specifically a form called "keyword in context". This is an indexing style used by scholars and which was once often available for some important some books and some websites. The idea is the concordance permits you to read or see a word that has been indexed in its original context, as this can often aid the reader to decide whether or not the reference is indeed what they want or need. Shown below is a screenshot of concordance displaying the context of the word "must" in a text corpus (taken from `https://bit.ly/3StMJhG`).



Concordance Hits 108

Hit	KWIC	File
1	come development. The National Orientation Agency must step up this campaign to mobilise Nigerians t	State Actor (
2	people with no common ideology other than "Buhari must go". It's just 22 days into 2018, So much	NSA Corpus
3	ria cannot survive another term of Buhari. #Buhari MustGo Poverty is one of the major reasons why	NSA Corpus
4	beside the wall * Owners of uncompleted buildings must clear them of undesirable persons or request	State Actor (
5	of cowardice nor timidity let alone impotence but must be sustained by courage, determination and co	State Actor (
6	ing corruption or those prosecuting corrupt cases must not engage in corruption it I bad to	NSA Corpus
7	must have self respect, self esteem. The children must see that daddy communicates with mummy. Don't	NSA Corpus
8	ce and development. If leadership fails, citizens must not fail and there lies the beauty and	State Actor (
9	to be part of this campaign. Our citizens must realize that the change they want to see	State Actor (
10	if we are saying the fight against corruption must be done within the ambits of the law,	NSA Corpus
11	m concerned, anybody who is accused of corruption must face the music. And mind you, some of	NSA Corpus
12	m concerned anybody that is accused of corruption must face the music. My only advice to government	NSA Corpus
13	ish poverty, insecurity and despair. "Our country must not be oblivious to concomitant danger around	State Actor (
14	ls of the campaign. "Terrorists and criminals must be fought and destroyed relentlessly so that	State Actor (
15	itizens' participation. Its language of discourse must be persuasive not agitated and abusive. It mu	State Actor (
16	us give peace a chance. The forthcoming election must not turn to war. This Nigeria belongs to	State Actor (
17	. There is no reward for it. Parents, fathers must instruct their children and ensure that they	State Actor (
18	wing, that discriminatory approach that the girls must go and wash plates while the boys are	NSA Corpus

Search Term ☐ Words ☐ Case ☐ Regex   Search Window Size
must   50
Kwic Sort: Level 1 1L, ☑ Level 2 1R, ☑ Level 3 2R

In the image the occurrences of the word "must" are shown in the context of sentences in which that word appear. Note that enough of the text before and after the word is shown. In this example, the concordance combines the references to "must" from several files (i.e. a collection of text files can be referred to as a "text corpus") at the right of each line. Other forms of concordance also provide a line number indicate the position of the line in the original text.

Our programs this semester will produce a text-file based concordance, where each run of the program focuses on a single text file for analysis. However, to keep coding simple, in this first assignment we focus on content that should appear in lines but without the referring to the line numbers in the output. (We will add this latter piece in later assignments.)

Building such an index requires, amongst other things, knowing which words must *not* be indexed. Definite and indefinite articles (*the*, *that*, *a*, *an*), conjunctions or modifiers (*and*, *but*, *or*, *not*), even prepositions (*on*, *at*, *under*, *between*, etc.) are examples of what need not be indexed.

In essence, this first assignment will require your program to transform some input text into a generated concordance output text. We will, however, include as part of the input those words to be excluded from the index listed. Consider the next page displaying an input file and the expected `concord1` output for that file. (These correspond to `in07.txt` and `out07.txt` from the test files provided to you for this assignment.)

```
1
''''
of
and
the
too
on
who
to
that
""""
that fortune
sense and sensibility
life of robert browning
the man who knew too much
legend of montrose
visit to iceland
orthodoxy
the mountains
on the track
ward of king canute
```

```
life of robert BROWNING
ward of king CANUTE
that FORTUNE
visit to ICELAND
ward of KING canute
the man who KNEW too much
LEGEND of montrose
LIFE of robert browning
the MAN who knew too much
legend of MONTROSE
the MOUNTAINS
the man who knew too MUCH
ORTHODOXY
life of ROBERT browning
SENSE and sensibility
sense and SENSIBILITY
on the TRACK
VISIT to iceland
WARD of king canute
```

The input file has a version number (for this assignment the version is 1), followed by a line with '''' (i.e four single quotes). There then follow the list of *exclusion words*, one word per line, with the list itself ended using a line with """" (i.e. four double quotes). Lastly the remainder of the file is made up of the lines-for-indexing of which the concordance will constructed for that file.

Each line in the output contains text based on one line-for-indexing of the input with exactly one word capitalized. When reading the output line-by-line (i.e., from top to the bottom) the capitalized words are in alphabetical order. Some lines-for-indexing *appear more than once in the output* as they have more than one indexed word. And as mentioned earlier in the assignment, the problem for A#1 has been simplified in that no line-number references appear in the output. (We will add this in subsequent assignments.)

Your task it to write `concord1.c` which constructs such output from such input.

Running the program

Your program will be run from the Unix command line. Input is expected from `stdin`, and output is expected at `stdout`. ***You must not provide filenames to the program, nor hardcode input and output file names.***

For example, assuming your current directory contains your executable version of `concord1.c`, (i.e., named `concord1`), and a `tests/` directory containing the assignment's test files is also in the current directory, then the command to transform the previous page's input into required output will be:

```
% cat tests/in07.txt | ./concord1
```

In the command above, output will appear on the console. You may want to capture the output to a temporary file, and then compare it with the expected output:

```
% cat tests/in07.txt | ./concord1 > temp.txt
% diff tests/out07.txt temp.txt
```

The same thing (i.e., producing output and comparing it with the expected output) can be combined into a one-liner:

```
% cat tests/in07.txt | ./concord1 | diff tests/out07.txt -
```

The ending hyphen/dash informs `diff` that it must compare the contents of `tests/out07.txt` with the input piped into `diff`*'s* stdin.

Exercises for this assignment

1. Write your program `concord1.c` program in the `a1/` directory within your git repository.
   - Read `stdin` line by line and processing each line appropriately.
   - Read and store exclusion words.
   - Read and store lines-for-indexing

- Create the output index by finding words in lines-for-indexing that are not in the exclusion words, and printing lines-for-indexing in the order of indexed words.
- Create and use arrays in a non-trivial array.
- Use the `-std=c11` flag when compiling to ensure your code meets the ISO/IEC 9899:2011 standard for C.

2. **Do not use `malloc`, `calloc` or any of the dynamic memory functions. Do not use regular expressions. AND DO NOT HARDCODE FILENAMES INTO YOUR PROGRAM! SORRY FOR SHOUTING!** See the "Simplifying Assumptions" section below for more supportive and positive guidance to help you with your implementation.

3. Keep all of your code in one file for this assignment. In later assignments we will use the separable compilation features of C.

4. Use the test files in `/home/zastre/seng265/a1/tests` (i.e., on the lab-workstation filesystem) to guide your implementation effort. Start with simple cases (for example, the one described in this writeup). In general, lower-numbered tests are simpler than higher-numbered tests. Refrain from writing the program all at once, and budget time to anticipate for when "things go wrong". There are ten pairs of test files.

5. For this assignment you can assume all test inputs will be well-formed (i.e., the teaching team will not test your submission for handling of errors in the input). Later assignments will specify error-handling as part of the assignment.

6. Use `git add` and `git commit` appropriately. While you are not required to use `git push` during the development of your program, you **must** use `git push` in order to submit your assignment.

What you must submit

- A single C source file named `concord1.c` within your git repository (and within its `a1/` subdirectory) containing a solution to Assignment #1. Ensure your work is **committed** to your local repository **and pushed** to the remote **before the due date/time**. (You may keep extra files used during development within the repository.)
- No dynamic memory-allocation routines are permitted for Assignment #1 (i.e., do not use anything in the `malloc` family of functions).
- You are permitted to declare arrays having a program scope, although you are encouraged to keep this to as small a number as is practicable given this may be your first time programming in C.

Simplifying Assumptions for Assignment #1

a) All input files will be concordance version 1 files.
b) At most 20 exclusion words will appear in an input file, and each word will be no more than 20 characters in length. It is possible that there are no exclusion words. Ignore any duplication in exclusion words.
c) At most 100 lines-for-indexing will appear in an input file, and each line is less than 70 characters in length.
d) Amongst all lines-for-indexing, there will be at most 100 distinct indexed words (i.e., if you count capitalized words an eliminate duplicates, there will be at most 100 such words).
e) Indexed words will appear at most once in any line-for-indexing.
f) All lines-for-indexing will be in lower case; words are separated by single spaces; and there is no punctuation.
g) Decimal numbers are treated as indexed words; use lexicographic (i.e., alphabetical) ordering for determining where line-for-indexing containing such a word should appear in the output.
h) If an indexed word appears in more than one line-for-indexing, print the lines in the order they appear in the input.

Evaluation

Our grading scheme is relatively simple and is out of 10 points. We may award grades within the categories below.

- "A" grade: An exceptional submission demonstrating creativity and initiative. `concord1` runs without any problems. The program is clearly written and uses functions appropriately (i.e., is well structured).
- "B" grade: A submission completing the requirements of the assignment. `concord1` runs without any problems. The program is clearly written.
- "C" grade: A submission completing most of the requirements of the assignment. `concord1` runs with some problems.
- "D" grade: A serious attempt at completing requirements for the assignment. `concord1` runs with quite a few problems.
- "F" grade: Either no submission given, or submission represents very little work.

Up to ten students may be asked to demonstrate their submitted work for A#1 to a member of the teaching team. Information about this will be communicated to selected students after the due date.