Programming Assignment #2
File Analysis and Small Search Engine

April 1, 2018

1 Objective

- To implement a tool that analyzes a text file and returns information about its content. It also allows a simple search for words in the file.

2 Submission Instruction

You are expected to submit using the online submission system using the upload file(s) link.

The submitted code file should be named a2.zip including

- **A proper inorder linked list implementation (inord_list.h + inord_list.cpp) that holds a structure appropriate for the intended usage**
- **Your program a2.cpp**

if your file has a different name, it will not be considered in the evaluation.
The files should be directly included in the zip file not in a folder inside the zip file.

Submission Deadline is To be Determined (sometime in the week following midterm)

- Missing the deadline == No Marks for this assignment
- Submit even if your code is partially working
- Write the code yourself. Plagiarism (code copying) in any of the assignments ==> **-10 Marks**

3 Assignment Overview

- In this assignment, we will use an in-order linked list to build a file indexing and (simple) search engine service.
- The program will be passed a text file name and a file containing commands.
- Your code is expected to
  - read the input file word-by-word. Separators such as , ; : ' ‘ & “ ” . [ ] { } ( ) should be discarded
  - each word is then inserted into the in-order list, along with the line number in which it exists in the file. First line in the file is line #1. Note: book, Book and BOOK are the same word (Case insensitive).
  - You should think of an efficient method to store the multiple occurrences of a word. The simplest is to insert multiple entries for the same word in the inorder list.
- The following actions will then be performed by the program. The file containing the commands, will have a set of commands. Each command is written in a separate line. The programs reads the commands line-by-line and executes them.
  1. command: wordCount –> Prints the number of words in the file
  2. command: distWords –> Prints the number of distinct words in the file
  3. command: charCount –> Prints the number of characters in the file (including spaces and newlines/carriage return)
  4. command: frequentWord –> Prints the word(s) that had occurred the most in the input file **excluding** common language constructs like (a, an, the,

in, on, of, and, is , are). If two or more words have the same number of repetitions, one should print all of them sorting them in an alphabetical order

5. command: countWord myword –> prints the number of myword occurrence in the file
6. command: starting mysub –> Prints all words that start with "mysub" followed by their number of occurrences. For example, if the command is "starting Te", it shall print all words starting with "Te", for example "Temperature" "Temperament".
7. command : containing mystr –> Prints all words that contain the string "mystr" followed by their number of occurrences. E.g. if the program is passed a command "containing al", it shall print words such as "shall" "all" in a table format (see below) /
8. command: search mystr –> Prints all words that contain the string "mystr" along with the line numbers in which they exist.

4 Typical Operation

- In all the following, a2.exe is assumed to be the name of your executable file
- Typically in the command prompt we write
  a2.exe <InpFileName> <CommandFileName>
- The files are always referenced to the current directory
- Your task is to **identify** the command and its parameters if applicable then **produce** the correct output.
- In all outupt lines, the default separator between words in the output is a single space unless otherwise specified.

4.1 wordCount

- input: a2.exe FileName CommandsFile
- wordCount exists in the CommandsFile
- output:
  1010 words

4.2 distWords

- input: a2.exe fileName CommandsFile
- distWords exists in the CommandsFile
- output:
  10 distinct words

4.3 charCount

- input: a2.exe fileName CommandsFile
- charCount exists in the CommandsFile
- output:
  50121 characters

4.4 frequentWord

- input: a2.exe fileName CommandsFile
- frequentWord exists in the CommandsFile
- output:
  Most frequent word is: book good

4.5 countWord

- input: a2.exe fileName CommandsFile
- countWord Nile exists in the CommandsFile
- output:
  nile is repeated 50 times

4.6 starting

- input: a2.exe fileName CommandsFile
- starting Te exists in the CommandsFile
- output:
  tease: 5 temporary: 10
  [separate word and its occurence (word record) by : and single spacce, separate

word records by tabs, sort words in alphabetical order. In the above output, an example of word record is "Tease: 5"]

4.7 containing

- input: a2.exe fileName CommandsFile
- containing al exists in the CommandsFile
- output:
  all: 5 shall: 2
  [separate word and its occurence (word record) by : and single spacce, separate word records by tabs, sort words in alphabetical order]

4.8 search

- input: a2.exe fileName CommandsFile
- search ne exists in the CommandsFile
- output: [put a colon after the word then separate word and lines occurrences by tabs, space between the word "lines" and the line numbers, spaces between line numbers, sort words in alphabetical order]

  neighbor: lines 5 62 123

  network: lines 12 17 278

  next: lines 110 114 119 1234

5 Sample Input File

The following is a sample CommandFile
wordCount
distWords
charCount
containing mega
countWord Biology
ending migo ==> should produce "Undefined command" error
search class

6 ERROR Handling

- You should check for the correctness of every command (e.g. number of arguments).
- You may use the following error messages to reflect the reason of not performing the command
  - "File not found"
  - "Undefined command"
  - "Incorrect number of arguments"
- **In case a search is performed on a string that is not found in the file, the output "Word not found" should be printed, e.g. in the starting, containing, and search commands.**
- Note that error messages are case sensitive.