HW 8 – Pick a Word, Any Word

Due Date: Midnight, Tuesday, Dec. 11th

Problem: A concordance is defined as “an alphabetical index of the principal words of a book or document, with a reference to the passage in which each occurs”. Creating a concordance from an existing document is a task requiring attention to lots of detail and an open-ended design since the exact size and contents of a source document are seldom known in advance.

Your assignment: Design, develop and test an Object-Oriented C++ program that prepares a concordance of <u>all</u> the words in any user-selected text file. Your finished result will be a file containing an alphabetical listing of all the words, and with each word, the number of times that word appeared in the document, and a list of the line numbers on which that word was found.

Discussion: Your program will need to read the contents of a user-selected text file, word-by-word, keeping track of line numbers, and keep a record of each word with all of its line numbers. Consider the following issues as you prepare your solution.

- Allow the user to select a data file from a menu of available files and save the results in a file named **“originalname_concordance.txt”** where **originalname** is the name of the input file without its extension.
- Create a user-defined class to manage each unique word from the file, including all the references to the word. This class should include at least the following features:
    - An instance variable to contain the word being cataloged. The word in this instance variable must be lower case and have all leading and trailing punctuation removed. Embedded hyphens, apostrophes, and periods must be left intact.
    - An instance variable to collect all the line numbers where this word was found in the file.
    - One or more constructors to initialize the instance variables.
    - One or more methods to manage the line numbers associated with the current word.
    - One method to format the word, its occurrence count, and its line numbers into a single string for output in this format: *word(OccurrenceCount): line, line, line, line, line*
- Create a user-defined concordance class to manage the entire collection of words from the data file. It should include at least the following features:
    - One collection of word objects (see previous class description).
    - One or more constructors to initialize the collection of word objects.
    - One or more methods to add and/or update the words and line numbers in the Concordance.
    - A means to maintain the collection in ascending order by Word, or a sort method to order the collection on demand.
    - One method to save the formatted contents of the Concordance to a user-specified text file.
- Create a menu-driven program that offers the user a list of available text files. The user will select a file from the menu.
- The program must have a single concordance object. This object must be reset each time an input data file is opened.
- Open the user-selected file and read every word in it, creating word objects and adding them to the concordance as necessary. For consistency, each word read from the file must be converted to lowercase and have all leading and trailing punctuation removed before being added to the concordance. Do not remove embedded punctuation, including periods, hyphens, and apostrophes.
- When the end of the data file has been reached and all words have been cataloged, save the formatted contents of the concordance object to a new file, named **“originalname_concordance.txt”** where **originalname** is the name of the data file without its filename extension.
- Allow the user to repeat this process for as many input files as they want.

Coding

- Each user-defined class must be defined within its own set of .h and .cpp files.
- Validate all inputs and do not proceed until valid values are entered.
- Format your source code according to the style guide presented in class.

Bonus

- Add a feature to the Concordance class to have it produce a second output file containing the data on only the 100 words with the highest occurrence counts. Name the file **"originalname_popular.txt"** where **originalname** is the name of the data file without its filename extension.

Data files: (these were all downloaded from http://www.gutenberg.org/)

- ModestProposal.txt – A Modest Proposal by Jonathan Swift
- Apology.txt – Apology by Plato
- WizardOfOz.txt – The Wonderful Wizard of Oz by Frank Baum
- MobyDick.txt – Moby Dick by Herman Melville
- ShortHistory.txt – A Short History of the World by H. G. Wells

Turn in a single zip file containing your source code in one or more ".h" and ".cpp" files. Name the zip file "First Last HW8", where "First Last" is replaced with your First and Last names.

Rubric:

Issue	**Poss.**	**Earned**
TURN-IN		
Source code submitted as a zip file containing only .h and .cpp files	3	3
Source code zip file named correctly	3	3
Source code zip file submitted via Canvas	3	3
Source code zip file turned in on or before the due date	3	3
BASIC FUNCTIONALITY		
Program presents a menu of appropriate options to the user	5	5
Program correctly validates the user's menu selection	5	5
Program reads all words from the selected data file,	10	10
Program converts each word to lowercase and removes all leading and trailing punctuation, while preserving embedded punctuation	15	15
Program records the line numbers of each occurrence of each word in the word object	15	15
Program creates and maintains only one word object from each unique word in the data file	13	13
Program places each word object into a single concordance object	20	20
Program saves the formatted concordance to a text file in ascending alphabetical word order	20	20
Program repeats till the user opts to quit	5	5
BONUS FUNCTIONALITY		
Program also saves the 100 most common words in the selected data file to a second text file	15	0
DEBITS – COMPILE AND RUN		
Program compiles with warnings	-10	0
Program compiles with errors	-10	0
Program crashes or runs with errors	-10	0
DEBITS – CODING TECHNIQUES		
No user-defined word class	-20	0
Word class has no local instance variables	-5	0
Words in each word object are not in lowercase or do not have only the leading and trailing punctuation removed	-5	0
Methods of the word class do not operate on local instance variables	-5	0
No user-defined concordance class	-20	0
Concordance class has no local instance variables	-5	0
Program does not annotate line numbers for each word correctly	-5	0
Program does not save the concordance data in ascending word order	-5	0
Methods of the concordance class do not operate on local instance variables	-5	0
Program does not reset/reinitialize the concordance object when the user selects a new data file	-5	0
Compares floating point values for exact equality	-5	0
DEBITS - CODING STYLE AND DOCUMENTATION		

Use of one or more goto statements	-10	0
Use of C-strings instead of string variables	-10	0
Call to main()to make the program repeat	-10	0
If statement without a matching else	-5	0
Switch statement not exhaustive and without default case	-5	0
Multiple returns from functions and methods	-5	0
Use of magic numbers	-5	0
Not using descriptive names for variables, constants, enums	-5	0
Not using descriptive names for functions, methods	-5	0
Identifiers not formatted correctly	-5	0
Improper indentation (body of every if, switch, do, while, for, try/catch)	-5	0
No comments in main()	-5	0
No comments in user-defined functions	-5	0
No comments in user-defined class, struct, or enum	-5	0
Total	**120**	**120**