Project 1 CS 136 Kutztown

Purpose: Review of material from CS 135; Intro to data structures (structs)
Points: 10 (drawings) + 25 (program)
Due: Drawings: 10 AM, Saturday, August 31, in D2L's Project1Drawings dropbox.
Program: 12 Noon on Friday, September 6, via the *turnin* utility.
Submit late at your own risk policy is in effect.
Description: Print the first 25 unique words found in a file, sorted, along with the number of occurrences of each word.

For your first project, you will write a program that initially prompts the user for a file name. If the file is not found, an error message is output, and the program terminates. Otherwise, the program inputs each token in the file, storing up to the first 25 unique ones found, and prints the tokens, sorted, each accompanied by the number of times it appeared, in a well formatted manner. To accomplish all this, do the following:

- Open the file – the user must be prompted and a file name input. DO NOT hardcode a file name.
- Read the words in the file, placing them in alphabetical order in an array of struct with capacity 25, where each struct has a string (the word) and an int (number of occurrences).
- Report the results

You must write appropriate functions for each task. At minimum, you must write a function for each of the following:

- Open the file:
  - Pass in an ifstream (must be passed by reference [what type of parameter is it?]). The function prompts the user and, if the file is found, it returns *true* and the ifstream will be a handle for that text file. Otherwise the function returns *false*, and the argument is assumed to be undefined.
- Populate the array of struct with the words in the file:
  - Pass in three reference arguments, the ifstream, the 25 element array of word/count structs, and the count of elements in use in the array. When the function completes, the array will hold the words in the file, ordered.
- Search for a token:
  - Pass in the array of struct (automatically by reference), its in-use count (by value), and a token (also by value). Return the index where the token was found, or, if not found, the index where failure could be established (see * below).
- Output results:
  - Pass in the variables that hold the file name (for printing purposes), the array, and its in-use count. Use this data to produce the desired output, which should be appropriately formatted.
- Increment the counter member of a struct that holds a word and its multiplicity:
  - Pass in a single struct; it will be an import/export parameter. Simply increment its counter member.

Opening the file and printing the tokens with multiplicity are both straight-forward. Populating the array is a bit more complex, as the list is to be maintained in order at all times.

*For each token input, the populate function will call the search function to obtain the index where the token is to be placed. The cases & actions for this are that the index returned by the search:

1. Equals the value of the array's in-use counter – Place new token at end
2. Is of a struct holding a token equivalent to the token searched for – Increment struct's counter
3. Is of a struct holding a token greater than the token searched for – New token goes in that index

Project 1 CS 136 Kutztown

You will create drawing(s) to demonstrate insertion. To present this effectively, show insertion of the contents of a file containing no more than 8 tokens, unordered, into a 5 element array of struct. Given that the array can be partially filled, show the process that will be carried out to insert tokens, where at least two appear multiple times. Show the array initially; this includes the array and its in-use counter. Then, show the progression of inserting each token, including the value returned by the search used by the insertion routine, until the file has been processed. Tokens must not be ordered alphabetically in the file.

Your drawing must show an instance of each of the three cases above for the value returned by the search, marked clearly and demonstrating what must be done visually and with a text description. It must be neat, with clear labeling of the cases and all actions. The sample data file should be displayed as well. Shoddy/incomplete drawings will receive commensurate grades or receive a grade of 0. No program will be graded without an acceptable drawing having been submitted first.

Notes:

- A word, or *token*, is whatever is read using the >> operator. A word followed by a period is distinct from the same word without the period. Words are also case sensitive, i.e. heLLo ≠ Hello.
- When a word is placed into the array of struct, its counter is initially 1.
- The purpose of creating drawings is to understand the problem completely before doing any coding. **DO NOT write any germane code until your drawings are complete and you fully understand how all cases are handled.**
- When the array reaches capacity, only repeats will be tabulated. New tokens will be discarded, i.e. only one of the possible results of the search function will result in updating anything in the array once it becomes full.
- While several functions are required, other functions should be written as well. One such function will handle Case 3 for the value returned by the search.
    - To keep the array sorted, write a function that works right to left to shift all elements greater than the new token one element to the right to create an open slot for the new data's struct. Be sure to update the in-use counter.
- Set up and test turnin **now**. You will not be permitted to submit late because your turnin wasn't set up or tested.
- **Work incrementally**. Your success in this course and the CS program may well depend upon your learning and practicing this concept.
    - Identify the basics and complete those first. Start by simply reading up to 25 tokens, placing them in the array and printing them. It doesn't have to be pretty. This way you know you can properly access and save data.
    - After completing each part of the project, test it before continuing to the next part. **Do not try to write the entire project without testing each facet as it appears to be complete.**
    - TEST YOUR PROJECT on acad (csit) BEFORE submitting!
- Your program must be well written, properly indented, and commented, including an id block. The instructor's website has information on proper documentation.

Deliverables: On the respective due dates:

- **D2L:** The drawing(s), well labeled, in a doc, pdf, jpeg, or tif, in the Project 1 Drawings dropbox
- **Turnin:** The program, named precisely **p1.cpp,** all lower case.

Minimum 2-point penalty for non-compliance to these directions.