Homework 4

100 Points

Structures and Strings

22B_H4A_Str_Move.cpp (c-strings)
22B_H4B_Ptr_Sort.cpp (find and fix errors)
22B_H4C_WordCount.cpp **Project: Text Analyzer: Word Count**

(See next pages)

Grading

Program 4A	– 15
Program 4B	– 15
Program 4C (index notation)	
1. Display welcome/farewell messages	– 5
2. Read words from file	– 15
3. Insertion Sort	– 15
4. Write report to file	– 15
5. Display words with the highest frequency	– 15
Self Assessment Report	– 5

Run the program as required and save the output at the end of the source file as a comment. Compress the source file, input and output files (if any), and upload the compressed file: 22B_LastName_FirstName_H4.zip

Note: Honors Assignment – Turn in two versions of Program 4C: one using index notation, the other one using pointers instead of indices.

Next Page

Project: Text Analyzer: Word Count

Write a program that parses a text file into words and counts the number of occurrences of each word. As you are reading the words, keep track of the number of distinct words, maximum length, and highest frequency. Read a new word then search for it: if found, add one to the counter; if not found, add the new word at the end of the list and initialize its counter to 1 (see a detailed example on the last page). Sort the array in alphabetical order (from 'a' to 'z') using the insertion sort algorithm. Save the sorted list of words to a file as shown below:

```
6 words
Maximum length: 7
Highest frequency: 125

  9 a
 21 boat
125 dream
 44 merrily
125 row
  5 stream
```

Display a list of high-frequency words as shown below:

```
125 dream
125 row
There are 2 high-frequency words.
```

You may assume that the text does not contain more than 100 words (i.e. declare an array of MAX_SIZE = 100) and that the frequency is not greater than 999 (for output formatting). Assume that the text files contain only letters, spaces and punctuation characters and that only the first letter in some words could be in upper case.

Run the program once using the following input files: **`song_row.txt`**, **`song_ten.txt`**, and **`test.txt`**(see below).

Create the input file **`song_row.txt`**, with the following data:

```
Row, row, row your boat,
Gently down the stream.
Merrily, merrily, merrily, merrily,
Life is but a dream.
```

Create the input file **`song_ten.txt`**, with the following data:

```
Ten green bottles hanging on the wall,
Ten green bottles hanging on the wall,
And if one green bottle should accidentally fall,
There will be nine green bottles hanging on the wall.
```

Create the input file **`test.txt`**, with the following data

```
The one two three waltz: one two three one two three step two
three step two three one two three one two three waltz
waltz waltz waltz waltz waltz. The end.
```

Next Page

The program will generate names for the output files by appending "**OUT**" at the end of the input file name, as shown below:
`song_row_OUT.txt`, **`song_ten_OUT.txt`**, and **`test_OUT.txt`**

Example: How to read data from file into an array of structures and keep track of the longest word's length and highest frequency.

```
Row, row, row your boat,
Gently down the stream.
Merrily, merrily, merrily, merrily,
Life is but a dream.
```

1. Read "Row,", convert it to "row", then insert it into the array:
   { {"row", 1} } // array has one item, highest frequency is 1, longest word is 3.

2. Read "row,", convert it to "row", since it is found in the array add 1 to its counter
   { {"row", 2} } // array has 1 item, highest frequency is 2, longest word has 3 characters.

3. Read "row", since it is found in the array add 1 to its counter
   { {"row", 3} } // array has 1 item, highest frequency is 3, longest word has 3 characters.

4. Read "your", since it is not found, add it at the end of the array
   { {"row", 3}, {"your", 1} } // array has 2 items, highest frequency is 3, longest word has 4 characters.

5. Read "boat,", convert it to "boat" then add it at the end of the array
   { {"row", 3}, {"your", 1}, {"boat", 1} } // array has 3 items, highest frequency is 3, longest word has 4 characters.

6. Read "Gently", converted to "gently", then add it at the end of the array
   { {"row", 3}, {"your", 1}, {"boat", 1}, {"gently", 1} } // array has 4 items, highest frequency is 3, longest word has 6 characters.

7. Read "down", then add it at the end of the array
   { {"row", 3}, {"your", 1}, {"boat", 1}, {"gently", 1}, }, {"down", 1} } // array has 5 items, highest frequency is 3, longest word has 6 characters.

8. Read "the", then add it at the end of the array
   { {"row", 3}, {"your", 1}, {"boat", 1}, {"gently", 1}, {"down", 1}, {"the", 1} }
   // array has 6 items, highest frequency is 3, longest word has 6 characters.

. . . and so on.

Next Page

Pseudo-code for the **readWords** function:

```
open input file (with validation)
set highest frequency to 0
set number of words to 0
set length of the longest word to 0
loop (not end of the file)
    read one word
    process word as needed
    if (word is not found in the list)
         add the new word at the end of the list
         set the counter of the new word to 1
         add 1 to the number of words
         update the longest length if needed
    else // found
         add 1 to its counter
         update the highest frequency if needed
    end if
end loop
close file
```