CS&E 1222

Lab 11 – Strings & Vectors Lab Assignment – 20 points

- ✓ The *lab* must be accomplished solely by you:
  - ➢ DO NOT look at anyone's code other than your own, including code from another's student in your section or another section of the course, or any third party source, e.g. the Internet
  - ➢ DO NOT share or copy anyone else's code for any graded assignment
  - ➢ DO NOT work in pairs or groups
- ✓ All cases of academic misconduct will be reported to the *Committee On Academic Misconduct* (COAM).

Setting up the Programming Environment

Effective commenting and tabbing will affect your grade. The "style" of your program should follow the style of the sample programs in the lecture notes. Also see the example code from Lab #1. Your program should have the file name, your name, creation and last modification dates and a brief description of the program in the comments. ***In addition, read the document on "Commenting" found in the Content tab on Carmen under "Tutorials"***.

1. At the Linux command line type `mkdir lab11`. This will create a new directory named **`lab11`**. Work out of this directory. In order to do that, type `cd lab11`. This changes the current working directory to the directory **`lab11`**.
2. If you have created the directory **`lab11`**, then just type `cd lab11`.
3. Copy the file **`freq_template.cpp`** by typing

```
cp /class/cse1222/9643/lab11/freq_template.cpp freq.cpp
```

4. Copy the file **`freq_solution.exe`** by typing

```
cp /class/cse1222/9643/lab11/freq_solution.exe .
```

Be sure to include ***9643*** (this is your course section indicator) and the period, ".".

Programming Assignment

Write a program in the file **`freq.cpp`** which reads text from the user and then computes the frequency of each vowel as well as the number of consonants that appear in the text. A vowel is one of the letters *a*, *e*, *i*, *o*, *u*, or *y*. Note that *y* can be considered to be a vowel or a consonant. We will count a lower case or capitalized version of a letter the same. The frequency of a letter in some text is the number of times it appears in that text, i.e its count. For example, the vowels in the following text "*I thrive on three tastes: Sweet, TANGY, and Salty!* " have the following frequencies: *a* occurs 4 times, *e* occurs 6 times, *i* occurs 2 times, *o* occurs 1 time, *u* occurs 0 times, and *y* occurs 2 times. The text has 23 consonants. Note that non-alphabetic characters are ignored and so not counted. Besides the *space* character, you can assume that the user will not type any other white space characters, such as a tab or other invisible characters such as control characters.

You will implement nine functions and the main procedure (see the code template in **`freq.cpp`**). Run **`freq_solution.exe`** to see an example of the program.

You will implement the algorithm described here and also described in the comments of the procedure main(). In short, you will copy over the characters from the input text (a *string*) into a C++ *vector* of characters, where you will include alphabetic characters only from the input text. Using C++ *vectors*, you will count the occurrence of each vowel and consonant and report these statistics to the user.

1. ***Read the lecture notes on "C++ Strings"and "C++ Vectors" before you start this lab!***
2. ***Write your code incrementally!*** Compile, run, and test your code *one function at a time*.
3. All function prototypes have already been provided for you and have been placed BEFORE the main procedure. Do NOT change the prototypes nor add additional prototypes.
4. All functions should be written AFTER the main procedure.
5. Each function should have a comment explaining what it does.
6. Each function parameter should have a comment explaining the parameter.
7. Do not add any more functions than those provided in the code template.

You must implement the following *algorithm* in the main() procedure to formulate your solution. *Follow these instructions* ***exactly****, including the specifications for the functions, to receive full credit*:

8. Five variables and one constant are defined in the procedure main(). You must use all of these in your solution to receive full credit.
   a. Variable ***input*** is a *string* that will hold the original input text typed by the user.
   b. Variable ***text*** is a *vector* of type *char* that will hold the alphabetic characters only of the input text.
   c. Variable ***vowels*** is a *vector* of type *char* that will hold the six vowels in lower case.
   d. Variable ***freqs*** is a *vector* of type *int* that will hold the count of each vowel encountered in the text. This vector directly corresponds with the vector in variable ***vowels*** and will have the same size as this vector (see below).
   e. Variable ***consonants*** will hold the number of consonant characters in the input text.
   f. Constant ***COLUMNWIDTH*** holds the width of your fields for displaying your formatted results.
9. For each function you will write below pay attention closely to how the input parameters are defined for the function.
10. Write the function init_vectors() to initialize two input vectors. You will add the six vowels to the vector in the first parameter. You will add six zeroes to the vector in the second parameter. Thus, if init_vectors(vowels, freqs) is called from the procedure main() then the variable ***vowels*** will contain characters {'a', 'e', 'i', 'o', 'u', 'y'} and the variable ***freqs*** will contain integers {0, 0, 0, 0, 0, 0}, assuming that both are initially empty. Use the *push_back* function provided in the C++ vector class to insert these values into the respective vectors.

Note ***freqs***[0] represents the number of *a*'s counted in the input, ***freqs***[1] represents the number of *e*'s counted in the input, and so on. Initially no counts have been determined yet.

11. Write a function read_text() that prompts the user for some text and returns the entered text as a *string*.
12. Write a function called create_list() that copies only the alphabetic characters from the input text (a *string*) to a C++ vector. First, write and test the following function:
    a. Write a function called is_alphabetic() that returns **true** if a character is an alphabetic character, i.e. *a-z* or *A-Z*; otherwise return **false**. Use the numeric ASCII value of the character to check if it falls within the two ranges of ASCII values for alphabetic characters in the ASCII table. See Lecture #3 on "Variables".

    Call this function in your solution to function create_list() to ensure that only alphabetic characters are inserted into the C++ vector.
13. Write a function called compute_vowel_freqs() that computes the frequency of each of the six vowels and returns the number of consonants found in the input text. The vowel frequencies are computed and stored in the last input parameter. Use a local variable to hold the count of the characters that are consonants. First, write the following two functions:
    a. Write a function called is_member() that returns **true** if a character appears in a given vector of characters; otherwise return **false**. Use a loop to compare the input character with each character in the C++ vector in the first parameter.
    b. Write a function called find_index() that returns the index location where the input character appears in the input C++ vector. Pass the vector of vowels as the first parameter. If the character does not appear in the vector of vowels then return **-1** (negative one).

    Loop through each character in the input text. If the character is a vowel (call function is_member() and pass it the vector containing the six vowels) then determine the index (call function find_index() and pass it the vector containing the six vowels) of this vowel in the vector containing the six vowels. Use this index location to index into the vector of frequencies and increment by one the associated count for that vowel. If instead the character is a consonant then increment your count for consonants (local variable) by one.

    Important note: when calling functions is_member() and find_index() apply the C++ *tolower*() function on the letter in the second input parameter of both functions. The C++ function *tolower*() takes a character as input and returns the lower case version of the character. Thus, it converts an alphabetic character to its lower case form. We must pass the lower case form of each character to the two functions is_member() and find_index() since we store the vowels in lower case form in our C++ vector holding the six vowels.

14. At this point variable ***freqs*** holds the count of each vowel and variable ***consonants*** holds the number of consonants found in the input text. Use the following two functions to display the frequency of the vowels:
    a. Write a function called display_characters() that displays each character from an input C++ vector of characters on a single line. Call this function with the vector holding the six vowels, i.e. *a*, *e*, *i*, *o*, *u*, *y*. Use the value in constant **COLUMNWIDTH** in six calls to *setw*() to format each field.

b. Write a function called display_freqs() that displays the counts in a C++ vector. Call this function with the vector holding the frequencies of the vowels. Again, use the value in constant **COLUMNWIDTH** calls to *setw*() to format the line of output.

Finally, display the number of consonants found in the input text.

15. TEST YOUR CODE THOROUGHLY. Determine boundary cases such as when the user just hits the enter key or types only non-alphabetic characters, spaces, leading spaces, etc. Your program must not have a run-time error.
16. Be sure to add the header comments "File", "Created by", "Creation Date" and "Synopsis" at the top of the file. Each synopsis should contain a brief description of what the program does.
17. Be sure that there is a comment documenting each variable.
18. Be sure that your if statements, for and while loops and blocks are properly indented;
19. Check your output against the output from the solution executables provided.

Submit Your Work

Important: Any program which does not compile and run will receive no credit!
If you are not sure what this means please ask your instructor.

Submit the file **`freq.cpp`** using the *Lab 11* drop box on Carmen. **DO NOT** submit the file **`a.out`**. **DO NOT** submit uncompleted work from *Quiz 11*. This will not be graded.