P4 Word Counter

- Due Oct 26, 2017 by 10am
- Points 100
- Available Oct 18, 2017 at 12am - Dec 1, 2017 at 11:59pm about 2 months

This assignment was locked Dec 1, 2017 at 11:59pm.

Program Description: This program we will be reading from a file and working with strings. Your program is required to ask the user for a file that it will open and then read. While your program is reading the file, it will be counting the total number of words by their lengths. When your program has read all of the file, then it will report to the screen the total number of word by their length, and the total number of words.

For example, if your program read the following file (demo.txt), it should give a report as shown below the file

File named demo.txt
Here is a test file that is full of some random words. I like icecream and pizza.

Output to the screen

```
   ~ Word Counter ~

What file would you like a report on (or type exit to quit): badname.txt
I'm sorry badname.txt could not be found.

What file would you like a report on (or type exit to quit): demo.txt


        ~Word Report for demo.txt
          Length:      Count:
            1           2
            2           3
            3           1
            4           7
            5           0
            6           3
            7           0
            8           1
           >8           0
Total words in the file: 17
```

```
What file would you like a report on (or type exit to quit): exit

Press any key to exit
```

Note: words. and pizza. both count as 6 letter words due to the .

As shown in the above example, you should ask the user for the file to be read, and if the file can not be opened then you should ask again until you can open a file or the user types exit to leave the program. Your program should count the number of words of length 1-8, and also keep a count for words over length 8 (>8). In addition, your program should display a total count for all the words (no matter what their length is) that were read from the file. Your output should look nice, so use some setw(n)'s.

demo.txt is to be used when testing your program. (do a right click and then select Save As)

infile.txt is to be used when you run your program to turn in the results. (do a right click and then select Save As)

You will need to put the files in your c++ project's directory.

You will need to use the fail( ) function to check to see if the input file failed to open correctly....meaning it could not find the file in the directory. Here is an example of using the fail function.

```
ifstrean fin;         // create the input file stream variable
string str = "myfile.txt";
fin.open( str );     // tries to open the file
if( fin.fail() )      // .fail() returns true if it could not open the file
    cout << "Unable to find the input file\n";
```

Remember: After you have used fstream variable (fin or fout, or whatever you call them) you will need to do one of the following:

Situation	**Needed Action**	**Example**

if you want to use the fstream variable again (after using it successfully or after it failed to open)	close and clear the fstream variable	**fin.close( );**
		fin.clear( );

if you are finished with the fstream variable (and don't want to use it again)	close the fstream variable	**fin.close( );**

You can declare (create) your fstream variable inside the while loop so that it is created each time the loop runs (this is the easiest solution).

Strings:

You will need to use the string object to be able to read the words from the file, and to be able to get a file name from the user. We have not talked about the string class yet, but using strings are pretty easy, and I think you can understand how to use them from the examples given below. What is a string? It is 0 or more char's. For example, these could all be stored as strings: "hello", "123", "F-47" "hello there". One thing to keep in mind when we are using the extraction operator (>>), is that it will stop reading when white space is encountered. For example:

```
string str;
cin >> str;        // user types: hello there
cout << str;       // hello is displayed
```

String Commands		
Situation	**Needed Action/ Function**	**Example**
Create a string variable		**string strVar;**
Reading input into a string variable	>>	**cin >> strVal;**
Comparing a string variable	= = and !=	**if (strVar == "hello") orif (strVar != "goodbye")**
Getting the length of a string variable	int length( )	**len = strVar.length( );**
Convert a string variable to a C String (as of C++^{11} not needed any more for the ifstream::open( ) function)	cstring c_str( );	**fin.open(strVar.c_str( ));**

Hints: Write the program a piece at a time. Write a few lines of code, and then test those lines of code. Use cout's to tell you about what is happening in the program. Ask me if you have questions...about the requirements or when programming.

Turn in: A paper printout of your source file (cpp) with the output from your program when it reads the **infile.txt** file. **Note**: First type the file name incorrectly and then type it correctly. Put the output below your code as a comment (between /* and */).

As always, you can't change your input or output files to make the program work.

Grading: I will be grading your program on the following:

- Did you follow the style shown in the class and book (indenting inside of a block)
- Did you include the Program Comments as shown below
- Did your output have a good spacing and spelling as shown in the sample program
- Did you use descriptive names for your variables
- Did you get the right output

- Did you use the input files I told you to use
- Did you avoid line wrap (when a line is too long and so it is wrapped around to the next line when printed)
- Did you staple all your papers together

Comments: Comments are a way of documenting a program (explaining who did what and how). All programs for the rest of the course are required to have the following header documentation and inline documentation to explain any tricky pieces of code.

```
////
// Name: Susie Programmer
// Section: A, B, or S
// Program Name: Hello World
//
// Description: A brief description of the program.  What does the
//  program do (not how it does it: for example, it uses loops)?  Does
//  the program get input?  What kind?  What information is output
//  from the program and to where (screen or file)
////

#include <…>

.......the rest of the program
```