ECE 71 – Introduction to Computational Programming

Professor Kriehn – Fall 2017

Code Due By: Midnight on Monday, October 23, 2017
Write-up Due By: Class on Tue/Wed, October 24/25, 2017

HOMEWORK #21 – Average Word Length

Write a program that computes the average word length (the average number of characters per word) for a file that contains some text. Characters associated with a word are defined to be composed of any alphabetical characters, the apostrophe character in the case of a contraction, and a single hyphen in the case of a hyphenated word. Words are separated by spaces or return characters. Whitespace characters, double dashes, punctuation, and other non-alphabetic characters (except those previously noted) should be ignored in the character count for a given word. A new paragraph (two return characters) should not increase the character or word count.

Specifications:
The text file is called "**`bob.txt`**", and can be found on the website for ECE 71. It contains the lyrics for Weird Al Yankovic's palindrome song, "**Bob**", in which every line is composed of a palindrome:

https://www.youtube.com/watch?v=JUQDzj6R3p4

Copy and paste the file into a "`bob.txt`" file within your project directory, in a manner similar to HW 20.

Use a function to read in each character one at a time from the file, following the rules noted above to determine whether the character should be counted toward a word or not. Only count characters that are part of words, and be sure to increase the word count only for a valid word. It will be helpful to keep track of the current character and previous character. The function should also work with the stream `cin` as an input stream, even though the function will not be called with `cin` as an argument in this program. Use the following function prototype:

```
void wordLength(istream &bob);
```

Print 3 columns to the screen. As characters are read in from the file, the first column will print every character in the file. The second column will print the current valid character count. The third column will print the current number of words. Separate each column with the tab character. After the entire file has been printed, calculate and print the average number of characters per word using 2 values after the decimal point.

The only purpose of the main function is to open the file, exit the program if the file fails to open, call the `wordLength()` function, and close the file.

As an example, if you execute the program with the following underlined inputs, the output will be:

```
~> main.o
B         Chars: 1       Words: 0
o         Chars: 2       Words: 0
```

```
b          Chars: 3       Words: 0

           Chars: 3       Words: 1

           Chars:    3         Words: 1

I          Chars:    4         Words: 1
,          Chars:    4         Words: 1
           Chars:    4         Words: 2
m          Chars:    5         Words: 2
a          Chars:    6         Words: 2
n          Chars:    7         Words: 2
,          Chars:    7         Words: 2
           Chars:    7         Words: 3

...

h          Chars:    610       Words: 188
o          Chars:    611       Words: 188
g          Chars:    612       Words: 188

           Chars:    612       Words: 189

           Average Chars/Word: 3.24
~>
```

Develop your I/O diagram and pseudocode, debug your code, and submit to the Grader Program.

ECE 71 – Introduction to Computational Programming

Professor Kriehn – Fall 2017

HOMEWORK #22 – File Parser

Write a program that numbers the lines found in a text file and keeps tracks of the character statistics of the file. The program should read text from the file and output each line to the screen, preceded by a line number. Also keep track of the number of paragraphs, the total line count, the total word count, the total character count, the alpha character count, the non-alpha character count, and the white space count. The sum of the alpha, non-alpha, and white space characters should be equal to the total character count in the file.

Specifications:
Use the "**`bob.txt`**" as the input file once again.

The function prototype for your computations should be:

```
void fileParser(ifstream &file);
```

The function should perform all necessary computations and print the results to the screen. Print the line number at the start of the line, right justified in a field of three spaces. Follow the line number with a colon, a single space, and the line of text.

The only purpose of the main function is to open the file, exit the program if the file fails to open, call the `fileParser()` function, and close the file.

As an example, if you execute the program with the following underlined inputs, the output will be:

```
~> main.o
  1: Bob
  2:
  3: I, man, am regal -- a German am I
  4: Never odd or even
  5: If I had a hi-fi
  6: Madam, I'm Adam
  7: Too hot to hot
  8: No lemons, no melon
  9: Too bad I hid a boot
 10: lisa Bonet ate no basil

...

 42: God! A red nugget! A fat egg under a dog!
 43: Go hang a salami, I'm a lasagna hog

Paragraph Count:           4
Line Count:               43
Word Count:              189
Character Count:         850
```

```
Alpha Character Count:   606
Non-Alpha Char Count:     49
White Space Count:       195
~>
```

Develop your I/O diagram and pseudocode, debug your code, and submit to the Grader Program.