CS 10 - Assignment 4

Collaboration Policy

Collaboration between students on programming assignments is **NOT** allowed under any circumstances - you may not even discuss your work with others; you may **NOT** get coding assistance, or copy code, from **ANY** outside source *(books, web sites, current or past students, your own previous submissions if you are repeating the course, etc.)*. We test every submission for copying and when we find it we treat it as flagrant academic dishonesty on the part of ***all*** parties involved, regardless of their roles as provider or consumer. The penalty is generally instant failure in the course, in addition to the usual sanctions applied by Student Judicial Affairs.

As you can see from the submission header below, we ask you to essentially take an oath that the code you submit is ***entirely your own work***.

At the same time, we certainly understand very well that you will frequently need help completing your programming assignment, so we provide channels where you may get that help in a way which will strengthen your programming skills: instructor and TA office hours, and the discussion forums where you can ask all the questions you want about the assignment, and even help others with the understanding that you have gained *(but not, of course, with any actual code)*.

Assignment Submission Instructions

Your assignment must be submitted as a ***simple text file*** named ***main.cpp***
Files in ANY other format (MS Word document, etc.) or with ANY other name (main, main.doc, main.txt, etc.) will ***not*** be graded.

Submit your work via the appropriate iLearn assignment link, making sure you use the correct link and **click on the attach button**

We strongly recommend that:

1. you submit at least *6 hours* before the deadline, even if you haven't completed the program - you can re-submit as often as you like, and we will only grade the final submission
2. once you have submitted your final attempt, go back and download it back to your system: then run it just to make absolutely sure that it really is the file you intended to submit!

You are budding professionals, and are expected to take full responsibility for abiding by the requirements of a contract.

The ***only reason we will ever accept*** for a missed deadline is if the system administrators inform me that either the iLearn system or the CS department servers were off-line for a significant period around the time of the deadline *(In which case we will probably have notified you of alternative procedures)*.

Remember to include the following header information at the top of your program
*(**DO NOT** REMOVE THE // at the beginning of each line, these tell the compiler to ignore these lines)*

```
// Course: CS 10 <quarter & year>
//
// First Name:
// Last Name:
// Course username: <enter the username you use to login in the lab>
// Email address: <enter your cs or UCR student email address here>
//
// Lecture Section: <e.g. 001>
// Lab Section: <e.g. 021>
// TA:
//
// Assignment: <assn1, hw2, lab3, etc.>
//
// I hereby certify that the code in this file
// is ENTIRELY my own original work.
//
// ================================================================
```

NOTE: This header MUST appear at the top of EVERY file submitted as part of your assignment (don't forget to fill in *your* details and remove the <> brackets!!).

Copy & paste this header into your file then update personal details.

Assignment Specifications

The ***Flesch Readability Index*** is a number, usually between 0 and 100, indicating how difficult the text in a document is to read. The index was invented by Flesch as a simple tool to gauge the legibility of a document without linguistic analysis.

Your Assignment

You are to write a program that calculates the Flesch Readability Index for an input text file. We've simplified the rules a little to match your programming abilities at this point.

a. *Count all words in the file.*
   A *word* is any sequence of characters delimited by whitespace, whether or not it is an actual English word.
b. *Cout all syllables in each word*.
   To make this simple, use the following rules:
   i. Each *group* of adjacent vowels (a, e, i, o, u, y) counts as one syllable.
      (e.g. the "ea" in "real" contributes 1 syllable, but the "e ... a" in "regal" counts as 2)
      - utilize the `is_vowel` function to determine if a character is a vowel
   ii. Each word has at least one syllable, even if the previous rule gives a count of 0 (i.e. even if there are no vowels in the word).
   iii. Do NOT use any special rules for an 'e' at the end of a word.
      So, for instance in both the word "Chile" and the word "while", our rules would count the final 'e' as a second syllable.
c. *Count all sentences*.
   A sentence is ended by a period, colon, semicolon, question mark, or exclamation mark.
d. *Compute the index, using this formula:*

```
index = 206.835 - 84.6 * syllable count / word count - 1.105 * word count / sentence count
```

e. *Finally, output the Flesch index of your text, rounded to the nearest integer.*
   (use the io manipulators `fixed` and `setprecision` for this step)

Header File

To assist you with the syllable count, I have provided a header file that contains the definition of a function named `is_vowel`. This function is passed in a character and returns `true` if the character is a vowel or `false` if it is not a vowel. Here is some code showing how to use the function:

```
string word = "regal";
char ch = word.at(0);
cout << ch;
if (is_vowel(ch))
{
   cout << " is a vowel." << endl;
}
else
{
   cout << " is not a vowel." << endl;
}
```

The above code should output: `r is not a vowel.`
However, if you replace `word.at(0)` with `word.at(1)`, the the output should be: `e is a vowel.`

To use this header file, `assn.h`, you need to first save it to the same directory that contains your source code, `main.cpp`. Then you need to include it similar to how you include the `iostream` library. At the top of your `main.cpp` file you should type:

```
#include <iostream>
#include <iomanip> // for fixed and setprecision
#include "assn.h"  // for is_vowel function

using namespace std;
```

Output Requirements

To more easily provide you partial credit, the following should be output by your program EXACTLY as it is shown here (with the values based on the particular input file).

```
Words: 1320
Syllables: 2062
Sentences: 53
Legibility Index: 49
```

DO NOT output anything other than this. **DO NOT** output any prompts to the user.

Input Requirements

We will utilize input redirection with this assignment. The input file should contain the text to calculate the Legibility Index on. We will utilize `cin` to take in the contents of the file as input.

Testing

File Contents	Words	Syllables	Sentences	Index
pfft.	1	1	1	121
aeyiou	1	1	1	121
Every man dies, not every man really lives.	8	14	1	50
87.5% of students are not sleeping enough!!!	7	11	1	66
a. e? i! o; u: y. b. c? d! f: z:	11	11	11	121
sdf sdf sdf.	3	3	1	119
bf.d bfd.	2	2	1	120

Example Run

Input file contents: 87.5% students cannot sleep!!!

```
[user@well ~]$ g++ solution.cpp
[user@well ~]$ ./a.out < input.txt
Words: 4
Syllables: 6
Sentences: 1
Legibility Index: 76
[user@well ~]$
```

Partial Credit

- Make sure you turn in code that compiles. All tests will fail if your program does not compile.
- If you can at least get your program to output the correct values for some of the counts but not all, please output the calculated ones and output a 0 for those that you did not calculate.
- If you do not get syllable count correctly calculated, use a 0 in place of the syllables count to calculate the readability index with just the sentence and word counts.
- Make sure you output all four values even if some are 0 due to not calculating.
- If some portion of your code causes compilation failure or runtime errors, comment it out.

Marking Guidelines

- Make sure your code compiles.
- Make sure your output keywords and spacing match ours exactly.

Basic Style Guidelines

- indentation - follow proper 2-4 space indentation as you go into loops or branches
- spacing - blank lines should be used to separate logic blocks
- no line wraps - no line of code should have more than 80 characters
- "meaningful" variable names
- comments - each logical block of code should have a brief explanatory comment