CIS*2500 Intermediate Programming

Assignment #2: Files and String Processing

1.0 The Problem and your task

The following index (reference: Rudolf Flesch, *How to Write Plain English*, Barnes & Noble Books, 1979) was invented by Rudolf Flesch as a simple tool to gauge the legibility of a document without using linguistic analysis.

1. Count all **words** in the file. A **word** is any sequence of characters delimited by white space or the end of a sentence, whether or not it is an *actual* English word.
2. Count all **syllables** in each word. To make this simple, use the following rules:
    - Each group of adjacent **vowels** (a, e, i, o, u, y) counts as one syllable (for example, the "ea" in "real" counts as one syllable, but the "e..a" in "regal" count as two syllables). However, an "e" at the end of a word does not count as a syllable. Each word has at least one syllable even if the previous rules give a count of zero.
3. Count all **sentences**. A **sentence** is a group of words terminated by a period, colon, semicolon, question mark, or exclamation mark. Multiples of each of these characters should be treated as the end of a single sentence. For example, "Fred says so!!!" is one sentence.
4. The **index** is computed by:
   index = 206.835 – 84.6 * (#**syllables** / #**words**) – 1.015 * (#**words** / #**sentences**) rounded to the nearest integer.

The index is a number, usually between 0 and 100, indicating how difficult the text is to read. Some examples for random material for various publications are:

Comic book	95
Consumer ads	82
Sports Illustrated	65
Time magazine	57
New York Times	39
Auto insurance policy	10
Internal Revenue Code in the U.S.	-6

The purpose of the index is to force authors to rewrite their text until the index is appropriately high enough. This is achieved by reducing the length of sentences and by removing long words. For example, the sentence:
The following index was invented by Flesch as a simple tool to estimate the legibility of a document without linguistic analysis.
can be rewritten as:
Flesch invented an index to check whether a document is easy to read. To compute the index, you need not look at the meaning of the words.

Input

Your program will read the text to be analyzed from a file. The filename is to be given as a command line parameter to the program. You will submit your program in three files: (1) `main.c` that will have the main function, (2) `fleschIndex.h` that will include all function prototypes, constants, macros and libraries and (3) `fleschIndex.c` that will have function definitions of all functions that are declared in `fleshIndex.h`. Your program will execute the code on a file by doing the following:

```
./fleschIndex <filename>
```

For example, if you have a file with an essay and the file was named *myEssay.txt* then you would do the following to find the Flesch index:

```
./fleschIndex myEssay.txt
```

Required functions

There are 3 required functions that your program must define and use (readFile, calculateFleschIndex and outputFormattedFleschScores). Prototypes of these functions are given below.

```
char * readFile ( char * filename)
```

This function takes a filename of a text file as input, opens and reads the contents of the file and returns the contents as a string (char *). **Note that you are required to use dynamic allocation, so that the returned string is on a heap.** For example, if the input file is test1.txt, then this function returns a char * that holds the following characters (given in red):

```
The red readymade dress was made for you!  It was going to be ready tomorrow.
What was the colour of the dress?  Oh, it was red!
```

You will find this text file (test1.txt) and 2 other test files (test2.txt and test3.txt) in the assignment folder on moodle. Use them for testing.

```
int calculateFleschIndex(int syllables, int words, int sentences);
```

This function takes the total number of syllables, words and sentences, calculates the flesch index and returns it.

```
void outputFormattedFleschScores(int syllables, int words,
                                 int fleshIndex, int sentenceCount);
```

This function prints the results on standard output (stdout) as the following:

1. The Flesch/legibility index that you have computed
2. The number of syllables in the input
3. The number of words in the input
4. The number of sentences in the input

It will have the following format:

```
Flesch Index = 87
Syllable Count = 10238
Word Count = 2032
Sentence Count = 193
```

Recommended functions

Besides the 3 required functions, you are recommended to define and use other functions – for example, functions for finding sentences, words and syllables in a given string. You may define and use any number of helper functions also.

2.0 Coding Guidelines, Assumptions and Testing

You must follow the style guide posted on moodle (carried over from CIS1300 and CIS1500).

Note that your program should read the entire file into memory and then do the counting and computation of the index.

You must do the following error checking / make the following **assumptions**:

- Your program will terminate with a meaningful error message if you cannot open or read the file given on the command line.
- Numbers (in the form of 2, 45, 865, *etc*.) will be counted as one word and one syllable.
- There will not be currency in the text (nothing of the form $12.50) and no floating point numbers.
- The only valid punctuation characters are the sentence termination characters: period, colon, semicolon, question mark, and exclamation mark. All other punctuation such as commas, #, @, *etc*. are to be **ignored**.

There are 3 text files (test1.txt, test2.txt and test3.txt) attached to the assignment on moodle that you may use to test your code – the final output when these files are used for testing is also attached as soln1.txt, soln2.txt and soln3.txt. Note that these are sample text files – we may use other files to test your code.

3.0 Compilation

Your program will be compiled using a makefile with the following flags:
`-std=c99 -Wall`

You must create a makefile, in addition to the program files. The main target must be called `fleschIndex.` Note your makefile is required to have a target called `clean`, that allows to remove all object files and the executable file (`fleshIndex)`.

Files required for compilation:

- `fleschIndex.c` — your function implementations. You must submit this.
- `fleschIndex.h` — your function definitions/prototypes, constants, library imports, student info, and declaration of academic integrity. You must submit this.
- `main.c` This file must contain a `main()`.

4.0 Submission

You will submit your assignment files (`fleschIndex.c`, `fleschIndex.h, main.c and makefile)` to the submission box for A2 submission folder on GitLab.

************Instructions for Gitlab – Start (repeated from Lab1 / A1)**************

Step 0: Computer setup – this is an optional step.
- make sure git is installed (https://git-scm.com/downloads)
- Mac users can use the terminal mode (I have tested it on my mac)
- Windows users can use powershell, WSL (windows subsystem for linux) or git bash.
- decide where cis2500 work will go and make a directory (e.g. mkdir CIS2500)
- cd to that directory from the terminal application

Step 1: Connect to the school server via noMachine or (portkey.socs.uoguelph.ca / linux.socs.uoguelph.ca).

Decide where cis2500 A2 work will go and make a directory (e.g. mkdir a2) if you need to.
cd to that directory from the terminal application

Step 2: From the chosen directory, now type:
git clone https://git.socs.uoguelph.ca/2500w21/<your username>/a2.git
At this point, you have a directory to work with on your local system.

Step 3: Do the work
- (remember to) cd to the new directory
- create the file that you are working on
- after first save, type: git add filename
ONLY ADD THE FILE ONCE!!! //do not do: git add .
Loop every 20-30 minutes:
git commit -am "write something here about what you just did"
Once per day:
git push // this is what stores local work back to the server.

To learn more about Gitlab, go to this link on moodle
https://moodle.socs.uoguelph.ca/course/view.php?id=169
************Instructions for Gitlab – End *******************

Your `fleschIndex.h` header file will contain a program header comment signifying your academic integrity. It will be of the following form:

```
/*
  Student Name: Firstname Lastname
  Student ID: #######
  Due Date: Mon Day, Year
  Course: CIS*2500

  I have exclusive control over this submission via my password.
  By including this header comment, I certify that:
   1) I have read and understood the policy on academic integrity.
   2) I have completed Moodle's module on academic integrity.
   3) I have achieved at least 80% on the academic integrity quiz
  I assert that this work is my own. I have appropriate acknowledged
  any and all material that I have used, be it directly quoted or
  paraphrased. Furthermore, I certify that this assignment was written
  by me in its entirety.
*/
```

5.0 Marking

- Programs that don't compile receive an immediate zero (0).
- Programs that produce compilation warnings will receive a deduction of 1 mark per unique warning (i.e. 3 'unused variable' warnings will only deduct 1 mark).
- Loss of marks may also result from poor style (e.g. lack of comments, structure)
- Programs that use goto statements receive an immediate zero (0)
- Programs that use global variables statements receive an immediate zero (0)