This is CS50

CS50's Introduction to Computer Science

OpenCourseWare

Donate (https://cs50.harvard.edu/donate)

David J. Malan (https://cs.harvard.edu/malan/)
malan@harvard.edu
(https://www.facebook.com/dmalan) (https://github.com/dmalan)
(https://www.instagram.com/davidjmalan/) (https://www.linkedin.com/in/malan/)
(https://www.reddit.com/user/davidjmalan)
(https://www.threads.net/@davidjmalan) (https://twitter.com/davidjmalan)

Readability

For this problem, you'll implement a program that calculates the approximate grade level needed to comprehend some text, per the below.

```
$ ./readability
Text: Congratulations! Today is your day. You're off to Great Places! You're off an
Grade 3
```

Getting Started

Open VS Code (https://cs50.dev/).

Start by clicking inside your terminal window, then execute `cd` by itself. You should find that its "prompt" resembles the below.

```
$
```

Click inside of that terminal window and then execute

```
wget https://cdn.cs50.net/2022/fall/psets/2/readability.zip
```

followed by Enter in order to download a ZIP called `readability.zip` in your codespace. Take care not to overlook the space between `wget` and the following URL, or any other character for that matter!

Now execute

```
unzip readability.zip
```

to create a folder called `readability`. You no longer need the ZIP file, so you can execute

```
rm readability.zip
```

and respond with "y" followed by Enter at the prompt to remove the ZIP file you downloaded.

Now type

```
cd readability
```

followed by Enter to move yourself into (i.e., open) that directory. Your prompt should now resemble the below.

```
readability/ $
```

If all was successful, you should execute

```
ls
```

and see a file named `readability.c`. Executing `code readability.c` should open the file where you will type your code for this problem set. If not, retrace your steps and see if you can determine where you went wrong!

Background

According to Scholastic (https://www.scholastic.com/teachers/teaching-tools/collections/guided-reading-book-lists-for-every-level.html), E.B. White's *Charlotte's Web* is between a second- and fourth-grade reading level, and Lois Lowry's *The Giver* is between an eighth- and twelfth-grade reading level. What does it mean, though, for a book to be at a particular reading level?

Well, in many cases, a human expert might read a book and make a decision on the grade (i.e., year in school) for which they think the book is most appropriate. But an algorithm could likely figure that out too!

So what sorts of traits are characteristic of higher reading levels? Well, longer words probably correlate with higher reading levels. Likewise, longer sentences probably correlate with higher reading levels, too.

A number of "readability tests" have been developed over the years that define formulas for computing the reading level of a text. One such readability test is the *Coleman-Liau index*. The Coleman-Liau index of a text is designed to output that (U.S.) grade level that is needed to understand some text. The formula is

```
index = 0.0588 * L - 0.296 * S - 15.8
```

where `L` is the average number of letters per 100 words in the text, and `S` is the average number of sentences per 100 words in the text.

Let's write a program called `readability` that takes a text and determines its reading level. For example, if user types in a line of text from Dr. Seuss, the program should behave as follows:

```
$ ./readability
Text: Congratulations! Today is your day. You're off to Great Places! You're off an
Grade 3
```

The text the user inputted has 65 letters, 4 sentences, and 14 words. 65 letters per 14 words is an average of about 464.29 letters per 100 words (because 65 / 14 * 100 = 464.29). And 4 sentences per 14 words is an average of about 28.57 sentences per 100 words (because 4 / 14 * 100 = 28.57). Plugged into the Coleman-Liau formula, and rounded to the nearest integer, we get an answer of 3 (because 0.0588 * 464.29 - 0.296 * 28.57 - 15.8 = 3): so this passage is at a third-grade reading level.

Let's try another one:

```
$ ./readability
Text: Harry Potter was a highly unusual boy in many ways. For one thing, he hated t
Grade 5
```

This text has 214 letters, 4 sentences, and 56 words. That comes out to about 382.14 letters per 100 words, and 7.14 sentences per 100 words. Plugged into the Coleman-Liau formula, we get a fifth-grade reading level.

As the average number of letters and words per sentence increases, the Coleman-Liau index gives the text a higher reading level. If you were to take this paragraph, for instance, which has longer words and sentences than either of the prior two examples, the formula would give the text an twelfth-grade reading level.

```
$ ./readability
Text: As the average number of letters and words per sentence increases, the Colema
Grade 12
```

▶ **Watch a Recording**

Specification

Design and implement a program, `readability`, that computes the Coleman-Liau index of text.

- Implement your program in a file called `readability.c` in a directory called `readability`.
- Your program must prompt the user for a `string` of text using `get_string`.
- Your program should count the number of letters, words, and sentences in the text. You may assume that a letter is any lowercase character from `a` to `z` or any uppercase character from `A` to `Z`, any sequence of characters separated by spaces should count as a word, and that any occurrence of a period, exclamation point, or question mark indicates the end of a sentence.
- Your program should print as output `"Grade X"` where `X` is the grade level computed by the Coleman-Liau formula, rounded to the nearest integer.
- If the resulting index number is 16 or higher (equivalent to or greater than a senior undergraduate reading level), your program should output `"Grade 16+"` instead of giving the exact index number. If the index number is less than 1, your program should output `"Before Grade 1"`.

Getting User Input

Let's first write some C code that just gets some text input from the user, and prints it back out. Specifically, implement in `readability.c` a `main` function that prompts the user with `"Text: "` using `get_string` and then prints that same text using `printf`. And remember, as you work through this program, that if you make use of any library functions, be sure to `#include` any corresponding header files.

The program should behave per the below.

```
$ ./readability
Text: In my younger and more vulnerable years my father gave me some advice that I'
In my younger and more vulnerable years my father gave me some advice that I've bee
```

Letters

Now that you've collected input from the user, let's begin to analyze that input by first counting the number of letters in the text. Consider letters to be uppercase or lowercase alphabetical character, not punctuation, digits, or other symbols.

Add to `readability.c`, below `main`, a function called `count_letters` that takes one argument, a `string` of text, and that returns an `int`, the number of letters in that text. Be sure to add the function's prototype, too, atop your file, so that `main` knows how to call it. Odds are the prototype should resemble the below:

```
int count_letters(string text)
```

Then call that function in `main` so that, instead of printing out the text itself, your program now prints the number of letters in the text.

The program should now behave per the below.

```
$ ./readability
Text: Alice was beginning to get very tired of sitting by her sister on the bank, a
235 letters
```

▶ **Hint**

Words

The Coleman-Liau index cares not only about the number of letters but also about the number of words in a sentence. For the purpose of this problem, we'll consider any sequence of characters separated by a space to be a word (so a hyphenated word like `"sister-in-law"` should be considered one word, not three).

Add to `readability.c`, below `main`, a function called `count_words` that takes one argument, a `string` of text, and that returns an `int`, the number of words in that text. Be sure to add the function's prototype, too, atop your file, so that `main` knows how to call it. (We leave its prototype to you!)

Then call that function in `main` so that your program also prints the number of words in the text.

You may assume that a sentence:

- will contain at least one word;
- will not start or end with a space; and
- will not have multiple spaces in a row.

You are, of course, welcome to attempt a solution that will tolerate multiple spaces between words or indeed, no words!

The program should now behave per the below.

```
$ ./readability
Text: It was a bright cold day in April, and the clocks were striking thirteen. Win
250 letters
55 words
```

Sentences

The last piece of information that the Coleman-Liau formula cares about, in addition to the number of letters and words, is the number of sentences. Determining the number of sentences can be surprisingly trickly. You might first imagine that a sentence is just any sequence of characters that ends with a period, but of course sentences could end with an exclamation point or a question mark as well. But of course, not all periods necessarily mean the sentence is over. For instance, consider the sentence below.

```
Mr. and Mrs. Dursley, of number four Privet Drive, were proud to say that they were
```

This is just a single sentence, but there are three periods! For this problem, we'll ask you to ignore that subtlety: you should consider any sequence of characters that ends with a `.` or a `!` or a `?` to be a sentence (so for the above "sentence", you should count it as three sentences). In practice, sentence boundary detection needs to be a little more intelligent to handle these cases, but we'll not worry about that for now.

Add to `readability.c`, below `main`, a function called `count_sentences` that takes one argument, a `string` of text, and that returns an `int`, the number of sentences in that text. Be sure to add the function's prototype, too, atop your file, so that `main` knows how to call it. (We again leave its prototype to you!)

Then call that function in `main` so that your program also prints the number of sentences in the text.

The program should now behave per the below.

```
$ ./readability
Text: When he was nearly thirteen, my brother Jem got his arm badly broken at the e
295 letters
70 words
3 sentences
```

Putting it All Together

Now it's time to put all the pieces together! Recall that the Coleman-Liau index is computed using the formula:

```
index = 0.0588 * L - 0.296 * S - 15.8
```

where `L` is the average number of letters per 100 words in the text, and `S` is the average number of sentences per 100 words in the text.

Modify `main` in `readability.c` so that instead of outputting the number of letters, words, and sentences, it instead outputs (only) the grade level as defined by the Coleman-Liau index (e.g. `"Grade 2"` or `"Grade 8" or the like`). Be sure to round the resulting index number to the nearest `int`!

▶ **Hints**

If the resulting index number is 16 or higher (equivalent to or greater than a senior undergraduate reading level), your program should output `"Grade 16+"` instead of outputting an exact index number. If the index number is less than 1, your program should output `"Before Grade 1"`.

Walkthrough

How to Test Your Code

Try running your program on the following texts, to ensure you see the specified grade level. Be sure to copy only the text, no extra spaces.

- `One fish. Two fish. Red fish. Blue fish.` (Before Grade 1)
- `Would you like them here or there? I would not like them here or there. I would not like them anywhere.` (Grade 2)
- `Congratulations! Today is your day. You're off to Great Places! You're off and away!` (Grade 3)
- `Harry Potter was a highly unusual boy in many ways. For one thing, he hated the summer holidays more than any other time of year. For another, he really wanted to do his homework, but was forced to do it in secret, in the dead of the night. And he also happened to be a wizard.` (Grade 5)
- `In my younger and more vulnerable years my father gave me some advice that I've been turning over in my mind ever since.` (Grade 7)
- `Alice was beginning to get very tired of sitting by her sister on the bank, and of having nothing to do: once or twice she had peeped into the book her sister was reading, but it had no pictures or conversations in it, "and what is the use of a book," thought Alice "without pictures or conversation?"` (Grade 8)
- `When he was nearly thirteen, my brother Jem got his arm badly broken at the elbow. When it healed, and Jem's fears of never being able to play football were assuaged, he was seldom self-conscious about his injury. His left arm was somewhat shorter than his right; when he stood or walked, the back of his hand was at right angles to his body, his thumb parallel to his thigh.` (Grade 8)
- `There are more things in Heaven and Earth, Horatio, than are dreamt of in your philosophy.` (Grade 9)
- `It was a bright cold day in April, and the clocks were striking thirteen. Winston Smith, his chin nuzzled into his breast in an effort to escape the vile wind, slipped quickly through the glass doors of Victory Mansions, though not quickly enough to prevent a swirl of gritty dust from entering along with him.` (Grade 10)
- `A large class of computational problems involve the determination of properties of graphs, digraphs, integers, arrays of integers, finite families of finite sets, boolean formulas and elements of other countable domains.` (Grade 16+)

Execute the below to evaluate the correctness of your code further using `check50`. But be sure to compile and test it yourself as well!

```
check50 cs50/problems/2023/x/readability
```

Execute the below to evaluate the style of your code using `style50`.

```
style50 readability.c
```

How to Submit

In your terminal, execute the below to submit your work.

```
submit50 cs50/problems/2023/x/readability
```