Readability

Implement a program that computes 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 and away!
Grade 3
```

Reading Levels

According to Scholastic, 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 grade reading level and a twelfth grade reading level. What does it mean, though, for a book to be at a "fourth grade reading level"?

Well, in many cases, a human expert might read a book and make a decision on the grade for which they think the book is most appropriate. But you could also imagine an algorithm attempting to figure out what the reading level of a text is.

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, to give a formulaic process 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 what (U.S.) grade level is needed to understand the text. The formula is:

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

Here, `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 from Dr. Seuss:

```
$ ./readability
Text: Congratulations! Today is your day. You're off to Great Places! You're off and away!
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. And 4 sentences per 14 words is an average of about 28.57 sentences per 100 words. Plugged into the Coleman-Liau formula, and rounded to the nearest whole number, we get an answer of 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 the summer holidays more than any other ti
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 eleventh grade reading level.

```
$ ./readability
Text: As the average number of letters and words per sentence increases, the Coleman-Liau index gives the text a higher readin
Grade 11
```

▶ **Try It**

Specification

Design and implement a program, `readability`, that computes the Coleman-Liau index of the 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, write code in `readability.c` such that when the user runs the program, they are prompted with `"Text: "` to enter some text.

The behavior of the resulting program should be like the below.

```
$ ./readability
Text: In my younger and more vulnerable years my father gave me some advice that I've been turning over in my mind ever sir
In my younger and more vulnerable years my father gave me some advice that I've been turning over in my mind ever since.
```

Letters

Now that you've collected input from the user, let's begin to analyze that input by first counting the number of letters that show up in the text. Modify `readability.c` so that, instead of printing out the literal text itself, it instead prints out a count of the number of letters in the text.

The behavior of the resulting program should be like the below.

```
$ ./readability
Text: 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 h
235 letter(s)
```

Letters can be any uppercase or lowercase alphabetic characters, but shouldn't include any punctuation, digits, or other symbols.

You can reference https://man.cs50.io/ for standard library functions that may help you here! You may also find that writing a separate function, like `count_letters`, may be useful to keep your code organized.

Words

The Coleman-Liau index cares not only about the number of letters, but also 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).

Modify `readability.c` so that, in addition to printing out the number of letters in the text, also prints out the number of words in the text.

You may assume that a sentence will not start or end with a space, and you may assume that a sentence will not have multiple spaces in a row.

The behavior of the resulting program should be like the below.

```
$ ./readability
Text: It was a bright cold day in April, and the clocks were striking thirteen. Winston Smith, his chin nuzzled into his breast in
250 letter(s)
55 word(s)
```

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 perfectly normal, thank you very much.
```

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 may count that 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.

Modify `readability.c` so that it also now prints out the number of sentences in the text.

The behavior of the resulting program should be like the below.

```
$ ./readability
Text: When he was nearly thirteen, my brother Jem got his arm badly broken at the elbow. When it healed, and Jem's fears of n
295 letter(s)
70 word(s)
3 sentence(s)
```

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

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"`.