CS 112 - Homework 3

Deadline

11:59 pm on **Friday, September 16**

Purpose

To answer questions related to C++ interactive input/output, file input, file output, and array basics, to practice a bit more with file input/output, and to practice a bit more with arrays

How to submit

You will complete **Problems 1, 2, 3, and 4** on the course Canvas site (short-answer questions on basics of C++ interactive input/output, file input, file output, and arrays).

For **Problems 5 onward**, you will create the specified `.cpp`, `.h` , and `.txt` files on the CS50 IDE, and then submit those to the course Canvas site.

NOTE: While I list the separate files you need to submit for each problem below, I am going to set up Canvas to *also* accept `.zip` files.

That is,

- you can submit each `.cpp`, `.h`, and `.txt` file to Canvas
- OR, if you prefer, you may compress your files to be submitted into a single `.zip` file and submit that `.zip` file to Canvas.

Problem 1 - 5 points

Problem 1 is correctly answering the "HW 3 - Problem 1 - Short-answer questions on C++ input/output" on the course Canvas site.

Problem 2 - 9 points

Problem 2 is correctly answering the "HW 3 - Problem 2 - Short-answer questions on C++ file input" on the course Canvas site.

Problem 3 - 8 points

Problem 3 is correctly answering the "HW 3 - Problem 3 - Short-answer questions on C++ file output" on the course Canvas site.

Problem 4 - 8 points

Problem 4 is correctly answering the "HW 3 - Problem 4 - Short-answer questions on C++ array basics" on the course Canvas site.

Problem 5 - function `get_size`

Consider a file of words structured as follows:

- Its first line should contain an integer, assumed to be the number of words in that file.
- Its remaining lines should contain that many words, assumed to have just one word per line.

For example, such a file might contain:

```
4
eagle
bagels
glaring
regally
```

As a small warm-up, and as a small helper function for Problem 7's program, write a function `get_size` that expects just a desired file name, assumed to have the structure described above. It has the side-effects of trying to open that file and read just the value on its first line as an integer, and it tries to return the integer it hopefully read.

For example, if you had a file `lookity.txt` in the current directory with the contents shown above, then:

```
get_size("lookity.txt") == 4
```

(It is fine to write this to be very trusting, and assume the given file is structured correctly; we have not covered exception handling yet.)

Submit your files `get_size.cpp`, `get_size.h`, `get_size_test.cpp`, and at least two `.txt` files (each with a different number of words) used in testing `get_size` in `get_size_test.cpp`.

Problem 6 - function `add_to_file`

As a second small warm-up, and as another small helper function for Problem 7's program, write a function `add_to_file` that expects a desired file name and a word to add to that file, has the side-effects of trying to open that file *for appending* and appending the word to be added to the end of that file followed by a newline, and returns the length of the word it attempted to append to that file.

For example, if you had a file `stuff.txt` in the current working directory, then:

```
add_to_file("stuff.txt", "moo") == 3
```

...and now `stuff.txt` should have the contents it had before this followed now by `moo` and then a newline.

(Reminder: you can APPEND to a file by calling the **two-argument** version of the `ofstream`'s `open` method, using a second argument of `ios::app`. For example:

```
  // opening my-log-file.txt for appending
  ofstream log_file_stream;
  log_file_stream("my-log-file.txt", ios::app);
```

)

Submit your files `add_to_file.cpp`, `add_to_file.h`, `add_to_file_test.cpp`, and at least two `.txt` files used in testing `add_to_file` in `add_to_file_test.cpp`.

Problem 7 - function guess_word_from_file

It is *not* the most elegant thing, but provided along with this homework handout is a function `rand_int` that expects a desired minimum integer and a desired maximum integer, and attempts to use the C++11 `random` library to return a pseudo-random integer in the range [desired minimum given, desired maximum given].

Write a `main` function in `guess_word_from_file` that does at least the following:

- asks the user for a file containing the words to choose from, assumed to be structured as described in Problem 5
- uses `get_size` to read the number of words in that file, and then declares an array of words of that size
- reads those words from the given file into the resulting array

- uses `rand_int` to get a pseudo-random choice of index from that array, and:
  - makes the word at that index the word-of-the-day, and
  - uses `add_to_file` to append the word to a file named `words_used.txt`
- asks the user to enter their guess of a word or to just type enter to quit
- while their guess is not the word of the day or an empty string, uses the function `guess_match` from Week 3 Lecture 2 to display to the user the user how close their guess was, and asks them to enter either another guess or enter to quit
- gives them an appropriate farewell message, either congratulating them on guessing the word or just giving them an appropriate farewell.

Optional variations

Here are **optional variations** you may make to the above, if you would like:

- You may ask the user if they would like to play again, and use `rand_int` to select another/more words-of-the-day and allow them to play again.
- You may limit the number of guesses a user can make (that is, only allow them a certain number of guesses to guess the word).
- You may ask the user where they would like their results written, and write to that file each of the user's guesses followed by `guess_match`'s result, and/or how many tries it took them to guess the word, and/or additional information or statistics as you would like.
  - (But this would be *in addition* to using `add_to_file` to append the word(s) used to `words_used.txt`.)
  - You could also append to the file they specify, rather than open it for writing such that its existing contents are deleted.
- You may use the Week 2 lab exercise's `five_letter_word` and/or `ask_for_word` as desired. (You might also want to create a variation of `ask_for_word` that uses `getline`, to allow the user to be able to just type enter to indicate wanting to quit.)
- (And further variations may also be fine, but ask me first if you would not be also meeting the minimum requirements given.)

Submit your `guess_word_from_file.cpp` and all of the `.cpp` and `.h` files for *all* of the helper functions it uses.