HW05 - A Wordle Refactor

In this homework, you'll be refactoring a poorly written, yet compiling and functional wordle game in C into a genuinely better, more testable program. Leveraging what you did and learned from in homework 4, we expect you to now:

- break-up the given `wordle.c` program into separate, logical functions.
  - include *REQUIRES/ENSURES* contracts.
  - the code should still compile and work as advertised!
- rename variables into meaningful, readable names.
- write comments explaining the logic that's happening in the code.
- test each function and any game logic in `test_wordle.c`.

Note: We don't expect you to work on this assignment over the official holiday break. The given deadline and expectations for this assignment already have considered the holiday.

Reminder: What are the basic tenets of wordle?

- **Five-Letter Word**: The target word is always five letters long.
- **Limited Attempts**: Players have a set number of attempts (usually six) to guess the correct word.
- **Feedback on Guesses**: After each guess, the player receives feedback on each letter:
  - The correct letter is in the correct position.
  - The correct letter is in the wrong position.
  - Incorrect letter.
- **Word List**: The game typically uses a predetermined word list, and players try to guess the target word based on their knowledge and the feedback provided.
- **No Time Limit**: There's no time pressure in the game; players can take their time to strategize and make educated guesses.
- **Logical Deduction**: Success in Wordle involves logical deduction based on feedback from previous guesses, helping players refine their subsequent attempts.

Learning goals

- Learn how to decompose software by breaking it down into logical parts, making it easier to understand, read, and maintain.
- Gain experience refactoring code (this is a real skill!).
- Use software design and testing to improve an implementation.

Tasks

You will be asked to perform the following series of tasks:

1. *Create your homework-5 repository*, with the GitHub classroom invite and clone that repository locally.

   Copy over anything useful from your homework-4 repository.

2. *Refactor* the code in `wordle.c`, include contracts, and implement tests.

   Currently `wordle.c` is a mess! It compiles; the game works on the command line. But, it's not easy to follow and will become a chore to maintain going forward.

   For this assignment, we expect you to break up `wordle.c` into logical units/functions, akin **to some** of those you came up with in homework 4. We assess that there are about 3-4 functions that can extracted from this large, single function.

   You'll add contracts to these *new* functions (you can stress test them in `test_wordle.c`). And, you'll write tests for these functions and around the general logic of the game, *without adding additional bugs*.

   - Getting started:

     Everyone should be able to compile and run C code. For compilation, just check to make sure you have `gcc` installed (which should be available by default):

     ```
     gcc -v
     ```

     The README.txt has instructions on how to compile, run, and test the given Wordle code. The designated test file, `test_wordle.c` will run the game given no modifications:

     ```c
     int main() {
       // Test 1
       test_me(0);

       // NOTE: Comment this out when you're ready to test your game!
       wordle_game();

       // If the test passes, print a success message
       printf("Test passed!\n");

       return 0;
     }
     ```

     Comment out `wordle_game()` when you're ready to actually test your refactored implementation.

     Additionally, `test_wordle.c` contains a small `test_me(0)` example (shown above), which can be run to demonstrate the use of *REQUIRES/ENSURES* contracts in C:

```c
int test_me(int x) {
  REQUIRES(x > INT_MIN);
  int res = x < 0 ? -x : x;
  ENSURES(res >= 0);
  return res;
}
```

Note: The word list for the game and contract assertions are already included for you.

3. *Re-diagram the implementation.*

   We want you to revisit your diagram from homework 4 and modify it to match the structure of your refactor.

   You can commit an image, pdf, etc to your repository for review.

4. *Reflect.*

   Update the given *reflection.txt* file, briefly discussing how closely your pseudocode (including pseudo-contracts and pseudo-tests) did or didn't line up with your refactor of the given code. Did your setup have more or less functionality than the game given? Also, comment on how your diagram changed from the previous exercise.

5. *Open a Pull Request (PR)* and **do not merge it**.

   For this assignment, you should check out your branch off of the main branch, and push up this branch consisting of commit(s). Once this is all ready, please open a Pull Request to the main branch and leave it there!

6. **Challenge**: *Add difficulty to the Wordle game* and test it's functionality.

   Add the ability to choose the difficulty of your game on input:

   - **1** is easy
   - **2** is hard(er)

   If the setting is easy, that's the game as it works already. If the difficulty is set to **2**, then, during a guess, you should not be able to move forward (i.e. get another guess) unless you get at least one matching letter (either in the word or by position).

   This is worth *20* additional points.

We expect that you'll be using the [C/C++][c-vscode] Visual Studio Code extension to help refactor your C programs. Being that this is written in C, feel free to use another extension or editor that best suits you.

Resources

1. https://marketplace.visualstudio.com/items?itemName=ms-vscode.cpptools

2. https://web2.qatar.cmu.edu/~mhhammou/15122-s23/lectures/20-cintro/writeup/pdf/main.pdf
3. https://realpython.com/python-wordle-clone/
4. https://www.cs.cmu.edu/~15122/handouts/lectures/21-cnum.pdf

Deadlines and Deliverables

Due Date: Tuesday, November 28, 2023 at 11:59 pm.

Deliverable: For this assignment, you have one deliverable.

1) You will either upload your GitHub repository URL to Canvas under assignment 5 (embedded via Gradescope as a PDF, with just your andrewID and link it) or DM staff over Slack.

You may use up to two (2) late days out of your total of four. To use a late day, simply message us on Slack to let us know that you will be using a late day. FYI, we are tracking late days, so please be aware of how many you have left.

Assignment Review

Because this is a new class, we are asking you to fill out a short survey to help us calibrate the homework. This survey is ungraded, but your input will be very valuable for us in improving the course both for this semester and for future years. Fill out a short survey to help us improve the course!

Grading

The total assignment is worth 100 points, but you can earn an extra 20 points by doing the *Challenge* question.

Item	Points
Refactor the code in `wordle.c` (including contracts)	50
Implement tests (`test_wordle.c`)	30
Re-diagram the implementation	10
Reflection	10
Challenge: Add difficulty to the Wordle game	+20
Total	**120**