Homework 6

Applying the TDD framework onto previously written code

October 4, 2019

Abstract

The objective is to give you experience promoting existing code written by someone else into the TDD framework (because it is very good to mention during job interviews).

The context of this problem is the game of battleship. This code can be found at `https://github.com/gabrieledcjr/Battleship/blob/master/Battleship/battleship.c`. The assignment is to take this code, and convert it into a form in C++, in which each kind of ship is a class, the board is a class, each function is a method, and there is a test with multiple test cases for each method.

Be sure to construct test cases for each function you build. Put thought into the test cases. The goal is to know that a function works, when it passes all of its test cases. One test case per function is usually not enough.

- Construct a sequence diagram for your project. You can draw it by hand and include a phone/photo, or draw it with a software tool.
- Use the test-driven development style. Document this by taking a new screen shot of your test code every time you make a new test (See Figure 1.), and of your production code every time you extend your production code. It could be a large number of screen shots. Be sure to run your code every time you add a few lines of test or production code. (See Figures 2 and 3.) Do not allow the number of errors to get large. Note that you can keep the code being converted outside of the scope of compilation, and introduce small parts of it into the ode being compiled, as if you were developing it.
- The program must display the configuration of the seas, including ships, after each move.
- The program must record the chosen move, whether it was a hit, a miss, or a redundant shot, and successive sea and fleet configurations for each turn in a log file. You can choose the file format, but you must document that, and adhere to what you describe.

Things to do:

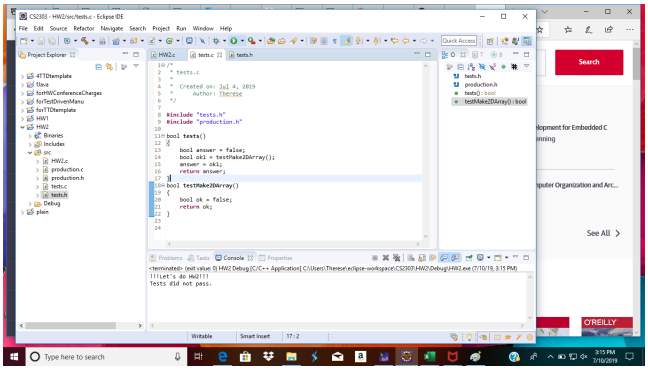

Figure 1: In the process of creating a new test.

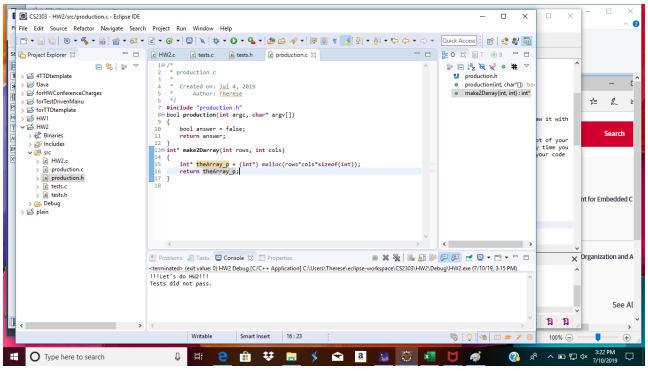

Figure 2: Starting some production code.

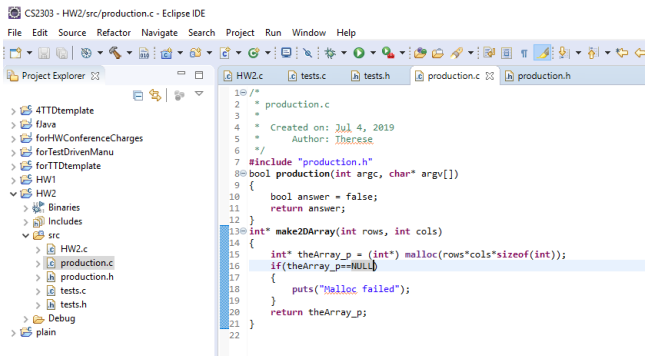

Figure 3: Adding more production code.

1. Either:

   (a) Make a C++ project from the Hello,World project.

   (b) Populate that project with at least one class for Test, and at least one class for Production, as well as at least one include file for tests and at least one include file for production.

   or use the starter code.

2. Create the sequence diagram extracted from the battleship code, and include the electronic file (diagram, screenshot or photo). Make sure your name appears within the sequence diagram.

3. Place function prototypes for all of your functions from the sequence diagram into one or more .h files.

4. As you work on the assignment, collect a sequence of screen shots showing how your test code, and your production code are growing.

5. Be sure to build and run often; do not allow errors to build up.

6. Show the sequence of sea and fleet configurations, including reporting a win.

7. Receive the values entered on the command line. Ask for values not provided in the command line. Note that the user is permitted to place their ships. Check for reasonable values. Negative number of turns implies no game should be conducted. Use these limit values in execution.

Grading

Criteria	Possible Points
Project that looks like starter code	20
Sequence diagram that reflects the problem statement	20
Documentation of code development (those screenshots) that clearly follows test-driven style, including adequate test cases	20
File of successive sea and fleet configurations	30
Correct treatment of command line arguments and other user entered data	10
Total	100