CS 170 | Spring 2017
Programming Assignment #3

Files due

- Wednesday, May 17
- 01:00am

Remember: The major point of this assignment is for you to learn how to submit your homework correctly. You should strive to follow all of the directions exactly as specified in the handouts. The syllabus that was handed out during the first day of class also contains important information on how to submit your homework. If you are still unsure of how to do something, you should ask for help, either from myself or from a tutor in the lab or from another student.

Information

This assignment will give you some practice with object-oriented coding (classes, objects, constructors, destructors, etc.) The task is to simply convert the previous WarBoats assignment (implemented as a procedural program) into an object oriented version. Because you can re-use almost all of your code from the previous assignment, this shouldn't take too long. The difficult parts have already been written.

Some Details (from the previous handout)

Instead of simply hard-coding size of the board at 10x10, we are going to allow the player to specify the dimensions of the board. (The board is the ocean.) We should be able to handle boards of any size (square and non-square). The program will only be limited by the amount of memory in the computer. We are also going to allow the player to specify the number of boats to place in the ocean. The only limit being the size of the ocean, since boats will not be allowed to overlap or leave the ocean. (Contrary to what Ferdinand Magellan's crew claimed, the world is flat, and if you go too far, you will fall off the end of it.)

All implementation will be placed in Ocean.cpp. You are not allowed to include any other files in Ocean.cpp. There are a couple of files included already, but you will not need any others.

New Stuff

The interface to the game is included in a header file named `Ocean.h`. This file contains all of the information that the client (the player) needs. A partial `Ocean.cpp` file is provided which includes the implementation of a couple of new methods.

New Method Details

Ocean(int numBoats, int xQuadrants, int yQuadrants);
Constructor - The client calls this to create an ocean (game board). Client specifies the dimensions of the ocean and the number of boats that will be placed in it. All private fields are initialized. No return value.

~Ocean(void);
Destructor - This method is responsible for deallocating anything that was allocated in the constructor. No return value.

Since 95% of the internal code is going to be the same as the previous assignment, it shouldn't take a lot of time to implement. It is just a matter of going through the existing functions and modifying them to be part of the *Ocean* class, instead of simply being global functions. Each private data member should be named with an "m_" at the beginning to make it clear that the variable is part of the class. You should follow this convention in future assignments when you will be required to create many of your own members. Remember to use **const** wherever appropriate. This includes both function parameters as well as member functions. Also, if you lost points on the previous assignment you should make sure and fix all of the problems so that you won't lose points again for the same thing (e.g. incorrect output, long lines, no comments, etc.).

Notes:

- Some students are forgetting that arrays in C/C++ are zero-based, meaning, if you have an array of 5 boats, you'd better only index that array with 0 through 4. Many students are

indexing that array using the BoatID (1 through 5), which is the cause of 99.9% of the problems. Remember, overwriting an array means your program is undefined (broken), which (unfortunately) usually means it will work for you on your machine with simple tests, but won't work for me when I test it. The same is true when doing bounds-checking to see if the boat or shot is outside of the array (ocean). So, please, please, please pay attention to the details when indexing an array.

- You are guaranteed that the user won't place more boats than numBoats.
- You are guaranteed that the ids will be in order from 1-numBoats.

Sample Command Lines:

GNU:

g++ -Wall -Wextra -Werror -O -ansi –pedantic -o gnu.exe driver-sample.cpp Ocean.cpp PRNG.cpp

Microsoft:

cl -W4 -EHa -Za -WX -Fems.exe driver-sample.cpp Ocean.cpp PRNG.cpp

Borland:

bcc32 -w -v -vG -ebor.exe driver-sample.cpp Ocean.cpp PRNG.cpp

After you run your Borland exe file you might have a file with an extension .cgl. Look in this file and see if you have a memory leak.

You are given two driver files. On this assignment, your makefiles should generate two executables called WarBoats.exe and WarBoatsBig.exe

Place all files into a zip file called CS170_<your.login>_3.zip.