Extra Credit Programming Assignment #1

This extra credit assignment builds off of Programming Assignment #2 (Object-oriented WarBoats). There are several modifications that need to be made to the code. All of the modifications were originally described in the practices. Here is an overview of the changes:

Default constructor, copy constructor and copy assignment operator
You must implement a default constructor, copy constructor, and copy assignment operator. The default copy constructor and copy assignment operator provided by the compiler are woefully inadequate and will cause your program to crash if you were to use them. You need to override them with your own implementations that perform a deep copy on the data. The provided default methods only perform a shallow copy. This is what the new default constructor should like like:

```
Ocean(int num_boats = 5, int x_quadrants = 10, int y_quadrants = 10);
```

Your current implementation of the constructor does not need to be modified to support this since we're going to use default arguments for the function. See the practice programming assignment called *warboats-ops* for details.

Comparison operator and insertion (output) operator
You need to implement the equality operator so that you can compare two Oceans. For our purposes, two Oceans are considered equal if they have the same dimensions, the same number of boats, and the same values in the grid array. Nothing else matters (e.g. how many shots have been taken). You also must overload the output operator so you can print an Ocean to the screen. See the output from the driver for details. (You should just take the original DumpOcean function and use that as a starting point.) See the practice programming assignment called *warboats-ops* for details.

`GetNumBoats()`
You need to make a public method called *GetNumBoats* that simply returns the number of boats in the Ocean. This will be used by the new driver to keep track of how many boats have been sunk. This is a trivial modification. See the practice programming assignment called *warboats-ops* for details.

Variable-length boats
Your new code needs to be able to handle boats of varying sizes. There is a new integer member of the *Boat* structure in WarBoats.h called length that will be used to store the boat's length. See the practice programming assignment called *warboats-varsize* for details.

Diagonal boat placement
In addition to boats being oriented vertically and horizontally in the Ocean, you will now need to handle boats placed diagonally. The enumeration in WarBoats.h has been updated to reflect this:

```
enum Orientation { oHORIZONTAL, oVERTICAL, oDIAGONAL_RIGHT, oDIAGONAL_LEFT };
```

See the practice programming assignment called *warboats-diagonal* for details.

Inter-boat spacing
Currently, boats can be placed anywhere inside the Ocean as long as they don't overlap with another boat. This means that two boats can "touch" by being adjacent to each other. We will have an option that allows the user to specify whether or not two boats can be adjacent to each other. The default constructor has been modified to accept a parameter that indicates this:

```
Ocean(int num_boats = 5, int x_quads = 10, int y_quads = 10, bool force_space = false);
```

If force_space is true, then you will make sure the boats have at least one intervening cell between them in the Ocean (grid). If it's false, the original behavior will occur, that is, two boats can be adjacent to each other.

There is much more information regarding these modifications on the practice pages for the course. Specifically, this extra credit assignment is comprised of these 4 practices which you can access from the web page for this assignment.

Practice assignment	Overview
warboats-ops	Additional operators, constructors, methods.
warboats-varsize	Allows boats to be of any size (length).
warboats-diagonal	Allows boats to be placed diagonally in the Ocean.
warboats-spacing	Allows boats to have space between them instead of touching another.

More on back →

What to submit

You must submit your header file and implementation file (Ocean.h and Ocean.cpp) in a .zip file to the appropriate submission page. Note that you are not submitting any other files.

Source files	Description
Ocean.cpp	The implementation file. All implementation for the functions goes here. You must document the file (file header comment) and functions (function header comments) using the proper techniques learned in CS120.
Ocean.h	The header file. You will need to modify this. **Absolutely NO implementation is permitted in this file.**

Usual stuff

Your code must compile (using the compilers specified) to receive credit. The code must be formatted as per the documentation on the website.

Make sure your name and other info is on all documents (paper and electronic).