

# Final Project - Battleship

For your final project, you will be writing your own code to simulate the game battleship. More details will be forthcoming, but for today (2/6/2018), you will have to draw the board that shows your board and your enemy's board, in the following style:

	1	2	3	4	5	6	7	8	9	10
A										
B										
C										
D										
E										
F										
G										
H										
I										
J										

The two boards should appear side by side.

```
void displayBoards(char [[10], char [[10]);
```

**Note:**When you display the board, you are also displaying the contents of the 2D arrays, even though initially they will be empty.

For your final project, you will be writing your own code to simulate the game battleship. This project will require you to use most of the concepts we have learned in the course.

Your program should have the following:

- Global constants storing the size of each of the 5 ships
- Global constant storing the size of the fleet
- A struct named `Point`, storing two ints that will have the row and column index of a particular cell in the grid
- A struct named `Ship`, that will store at least 4 variables.
  - The name of the ship
  - The size of the ship
  - The current hitcount of the ship
  - A vector of `Point` structs that will store the indices of each of the cells in the grid the ship occupies.
- A struct named `PlayerBoard` that
  - Stores the 10 x 10 board (char) for that particular player
  - Stores an array of `Ship` structs of size `FLEET_SIZE`
- A `setShip` function that takes in a `PlayerBoard` object by reference and an int variable that stores the index of the ship that is currently being placed, and places the ship onto the board. The `setShip` function calls the `getValidShipInfo` function to determine which spots on the board the ship will occupy.
- A `getValidShipInfo` function that takes in four parameters by reference, two ints holding the row and the column of the starting coordinates of the ship, a char that will hold the orientation of the ship (horizontal or vertical), and the `PlayerBoard`. The function will also take a fifth argument by value which is the index of the ship being placed. The function will prompt the user for the starting row and column coordinates of the ship which the user will enter as: letter number. These will then need to be converted to the proper row and column index of the array. The function will also prompt the user for the horizontal or vertical orientation of the ship. The function will be responsible for performing the error checking so that a valid ship placement is received from the user. The function will also call the function `spaceOccupied` to determine if any of the spaces the ship would take up if placed on the board are currently occupied.
- A `spaceOccupied` function that takes in the `PlayerBoard` object, an int for the row and col placement of the ship, a character for the orientation, and the ship size. This function returns true if the placement of the ship would overlap an already existing ship placement or false if the space is not occupied.
- An `initBoard` function that takes in two `PlayerBoard` objects by reference, and calls the `setShip` function for each ship in the fleet. After each ship is placed on the board the boards should be displayed.
- An `initFleet` function that takes in a `PlayerBoard` object as a parameter and initializes all the ships in the fleet with the appropriate information. For example, the name and size of the ship should be initialized within the function.

The rest of your implementation is up to you. Notice in the sample file you will have to do some input validation.

#	Class of ship	Size
1	Carrier	5
2	Battleship	4
3	Cruiser	3
4	Submarine	3
5	Destroyer	2

**Submission:**

You will demo this assignment to the instructor on the last day of class. June 11, 2018