Create a multiplayer battleship game using TCP as a communication protocol between 2 game clients. Keep in mind, either may need to act as a server to the other.

Prior to connection, each client will implement functionality to set ships. The number of ships will be as follows, 1 ship that is 2 grid positions long and 1 ship that is 4 positions long. Ships can be positioned in a horizontal or vertical alignment.

The grid scale for each client will be 10x10.

The game protocol will be as follows,

Client A makes connection to Client B and provides name. Client B will act as a server in this scenario.

Client B will then provide name to Client A.

Connecting client, Client A in this scenario, will always go first and will send coordinates to bomb in the format *x,y*. Client B will indicate whether one of its ships was hit by the bomb with either a 0 for no hit or a 1 for hit.

Between rounds clients should be presenting to the user a view of the enemy waters showing unknown grid locations, grid locations that have been hit, and grid locations that were misses.

When a client has had all of their battleships sunk, they will indicate to the other client that the game is over by sending *-1,-1*.

Each client will then reset ready to either connect to or receive a connection for another game. This includes potentially resetting ship locations.

Since all clients will follow the same protocol, you should be able to connect with and play your game against other class member’s implementations.

You may receive up to 3 extra points on your final exam grade if you implement a graphical user interface, but you may use a command line interface if you do not want to receive bonus points.

**Grading:**   
Correctness: You can lose up to 20% if your solution is not correct   
Quality: You can lose up to 20% if your solution is poorly designed   
Testing: You can lose up to 20% if your solution is not well tested   
Explanation: You can lose up to 40% if you cannot explain your solution during the grading session