EECS 338 Final Project

Clarinda Ho (cqh), Jason Shin (jjs270)

Concept

Create a 2-player battleship game that uses sockets (server and client). The server player is allowed to set the board size and the number of ships for each type.

Map

```
- Legend
        - B = Ship
        - X = Hit
        - _ = Empty
        -   = Miss

- Map View: left is what player sees, right is what opponent sees
  _ _ _ _ _   _ _ _ _ _
  _ _ _ _ _   _ _ _ _ _
  _ B X B _   _ _ X _ _
  _ _ _ _ _   _ _ _ _ _
  _ _   _ _   _ _   _ _
```

Initialization

```
- Printing out instructions at the beginning

- Configuration method
        - How large the board will be (max 20x20), how many ships to generate, etc.

- Creating the data structure to store the map (2D Array)
        - Each player will have two -- one for their map and one for their opponent

- Populate the board
        - Each player will choose where they will place the ship and the direction to orient the ship
        - Format: (Coordinate, Direction) i.e. 4 A EAST
```

Running

```
- Reading user input for two players
        - Format: Coordinates (Number, Letter) i.e. 4 A

- On Miss:
        - Show the miss on the map for the person shooting
        - Show the miss on the map for the defending player

- On Hit:
        - Check to see if ship is sunk
        - Keep track of ships in a struct, decrement a value representing how much health is left
        - Show the hit on the map for the player shooting
        - Show the hit on the map for the defending player

- Check to see if total health is 0
        - If total health is 0, end the game
        - If total health is not 0, switch turns
```

Design Document

```
- Files
        - battleship_client.c
                - Player 2 of battleship game
                - Client side of socket
        - battleship_server.c
                - Player 1 of battleship game
                - Server side of socket
                - Responsible for setting up the game (i.e. board size, number of each type of ships)
        - makefile

- Major Data Structures
        - Struct ship
                - int health: health points of the ship
                - int x[5]: x position of the ship
                - int y[5]: y position of the ship
        - 1D ship array
                - Contains all the ships of the same type on the player's board
                - One array for each type
        - 2D int array map
                - Contains the current state of the board
                - Values
                        - 0: empty
                        - 1: ship
                        - 2: hit
                        - 3: miss
                - When the map is printed, the numbers will be converted to corresponding character mentioned above in the map view
        - Socket & Server socket
                - Used for communication of moves between the two players
                - Used for communication of whether ships were hit between the two players

- Console Output
        - Sample player 1 output can be found in file 'sampleServerOutput.txt'
        - Sample player 2 output can be found in file 'sampleClientOutput.txt'
```

Team Member Responsibilities

```
- Coded using paired programming techniques
- Clarinda
        - Configured the server socket for battleship_server.c
        - Wrote configureBoard() method
        - Wrote the checkValidPos() method
        - Worked on the ship struct
        - Worked on turn(), attackTurn(), and defendTurn() methods
- Jason
        - Configured the socket for battleship_client.c
        - Wrote setupFromServer() method
        - Wrote the chooseShipPositions() method
        - Worked on the ship struct
        - Worked on hitShip() and related methods
        - Worked on isGameOver() method
```