**MAIN**

Displays Title

Gets player name

Creates Game object

Calls Game::gamelogic()

**GAME CLASS**

**Game::Game(\*)**

The constructor creates two Player objects

**Game::gamelogic()**

Calls Player::randomBoard() or Player::setBoard()

Sets level (Easy or Hard) and cheatmode (On or Off)

Enters game event loop

Generates random computer guesses or calls Game::AI()

Gets user input

Calls Player::displayBoard(\*)

Calls Game::playerTurn(\*)

Calls Player::endGame()

If Player::endGame() returns false

Calls Game::computerTurn(\*)

Calls Player::endGame()

Continues loop

If Player::endGame() returns true

Ends loop

**Game::playerTurn(\*)**

Calls Player::getPlayBoard(\*) to check for hit or miss

If hit, displays

Calls Player::markOppBoard(\*)

If a ship is sunk, displays

Calls Game::reveal()

If miss, displays

Calls Player::markOppBoard(\*)

**Game::computerTurn(\*)**

Calls Player::getPlayBoard(\*) to check for hit or miss

If hit, displays

Calls Player::markOppBoard(\*)

Calls Player::markPlayBoard(\*)

If a ship is sunk, displays

If miss, displays

Calls Player::markPlayBoard(\*)

**Game::reveal(\*)**

Loops through computers playboard using Player::getPlayBoard(\*) to find a match to its argument

If it finds a match, it calls Player::markOppBoard(\*)

**Game::AI(\*)**

Finds first instance of a ship using Player::getPlayBoard(\*)

Divides the board into 9 different areas using selection statements

Selects 9 grid positions based on grid area and randomly picks one of these until it finds a ship or a blank

**Game::endReveal(\*)**

Loops through players ships data member to find those ships yet to be sunk and calls

Game::reveal()

**PLAYER CLASS**

**Player::Player()**

The default constructor, creates the computer Player object

Sets the computers board

Sets the computers ships data member and name

Calls Player::randomBoard()

**Player::Player(\*)**

The constructor for the players Player object

Sets the players board

Sets the players ships data member and name

**Player::displayBoard(\*)**

Displays a header based on its argument

Displays the game board

**Player::setBoard()**

Enters loop

Gets user input

Calls Player::displayBoard(\*)

Divides the board into 9 different areas based on ship size and user input and offers valid input options

Calls Player::checkBoard(\*)

If Player::checkBoard(\*) returns true, displays invalid selection

If Player::checkBoard(\*) returns false

Calls Player::changeBoard(\*)

Calls Player::displayBoard(\*)

If user keeps, next ship

Else repeats above

**Player::randomBoard(\*)**

Enters loop

Generates random grid positions

Generates random allowable directions based on grid position and ship

Calls Player::checkBoard(\*)

If Player::checkBoard(\*) returns false

Calls Player::changeBoard(\*)

Else continues loop

**Player::changeBoard(\*)**

Changes the player board of the object, given the grid positions and ship as arguments

**Player::checkBoard(\*)**

Checks to see if any ships are in the direction of a potential ship placement

Returns true if a ship is in the way of a placement

Returns false otherwise

**Player::endGame()**

Checks the ships data member for the end game condition

Returns true for game over

Returns false otherwise

**Player::getPlayBoard(\*)**

Returns the playboard data member based on grid positions as arguments

**Player::getOppBoard(\*)**

Returns the oppboard data member based on grid positions as arguments

**Player::markPlayBoard(\*)**

Changes the playboard to the character given as an argument, at the grid position given as argument

**Player::markOppBoard(\*)**

Changes the oppboard to the character given as an argument, at the grid position given as argument

**Player::getShips(\*)**

Increments the ships data member by one at the index given as argument and returns it

\*Validation - Input validation throughout the program has been done with a view to prevent infinite loops or dead ends. This has been achieved through using cin.get() to get the first character entered and then flushing the buffer using cin.sync(). This is done for all user input in the program. Thus a user may enter multiple characters and only the first one entered will be read.

\*Cheat Mode - In cheat mode , the computers fleet is displayed but not changed as I use the computers board for the reveal functionality.

\* For best display, play with a minimized screen