/\*

gameType = 1 or 2 (1 = able to fire at only 1 target per turn, 2 = able to fire at multiple)

gameOver = 0, 1, or 2 (0 = game in progress, 1 = player1 won, 2 = player2 or computer won)

currentPlayer = 0 or 1 (0 = player1 or 1= player2)

target = 0 or 1 (0 = player1 or 1= player2 or computer)

playerStatus[2][6]

row = player1 or player2

column 0 = num of times player can fire (controlled by gameType and set at beginning of game)

column 1-5 = the status/health of player’s ships (initialized as [1] = 2, [2] = 3, [3] = 3, [4] = 4, [5] = 5 at the beginning of the game)

If any of the ship health equals zero, Then the ship is sunk so the user is shown the ship that was sunk, Else no change

computerStatus[6] = same as the columns above

attackLoc[5][2] = 5 rows (max num of attacks), 2 columns (X/Y Grid)

hidePlayer1 & hidePlayer2 = 0 or 1 (0 = show ships, 1 = hide ships, initialized at beginning of game and changed after every turn)

player1Grid[10][10] & player2Grid[10][10]

Each player can only see the entirety of their OWN grid

Each player can only look at the hits and misses of the opponents grid (the opponents grid will ALWAYS be HIDDEN from the player)

player1ShipLoc[5][2] & player1ShipLoc[5][2] = 5 rows (ships), 2 columns (X/Y Location)

void showGrid (grid[10][10], hide){

show grid

If hide == 0, Then show ships, else hide ships

show hits and misses (by opponent)

}

void getInput (currentPlayer, playerStatus, target, attackLoc){

noInput = true

do{

1) User decides to choose targets

A) get input(s) from GUI (buttons/switches)

B) get input(s) from keyboard as a string (optional, adjustable setting for player, will call a separate function and will alter the GUI state/ removes buttons and has text box instead)

C) Check that user chooses valid target(s)

D) Allow user to deselect the target (by clicking target/button again, keyboard setting will have menu to deselect target(s) for the user to choose)

2) User decides to pause the game/enter pause menu

Allow user to change settings (like volume, BGM, SFX, exit game, etc.)

Allows user to end/exit the game

3) User decides to fire at chosen targets

Check if target(s) are chosen, Then noInput = false, Else no change

} while (noInput is true)

return attackLoc[5][2]

}

void fire(currentPlayer, playerStatus, target, attackLoc, gameOver){

for j in range(playerStatus[currentPlayer][0])

}

­­

\*/

Battleship Game Modes

1931 Salvo Edition:  
In the modern Milton Bradley rules for *Battleship*, *Salvo* is listed as a variation "for more experienced players", with the following rule(s)

* players target a specified number of squares at one time, and all squares are attacked simultaneously. The number of shots a player can fire each turn may either:
  + be equal to the number of ships that the firing player has remaining
  + be fixed at five for the whole game
  + be equal to the size of the player's largest undamaged ship
* The receiving player may either call
  + the result of each shot in turn
  + or simply announce the number of hits and misses (ex: "two hits and three misses"), leaving their opponent to work out the consequences of the salvo
* One variant of Battleship allows players to decline to announce that a ship has been sunk (so the opponent takes further shots to confirm that an area is clear)
* Another house rule allows a player to move 1 of their ships to a new, UNCALLED location every fourth or fifth move