Our program is meant to create a semi-realistic battleship game against the computer, with difficulties of the user’s choice. When the game first starts two different windows pop-up, the players and the computers. The player is able to choose to play a ship horizontally or vertically as long is it fits within the board on their window and doesn’t overlap with any other ship. The place that you click on the board is where the top left part of the ship will be placed, which is done to have a consistent method of placement. After the user is done placing their ships, the computer will randomly place their ships following the same rules as the user. The actual game starts, and the player can fire by clicking on any open tile. If it is a hit, the a red circle will appear on that tile. If a miss, a blue circle will appear. If a player attempts to fire upon a tile with any circle on it, and therefore has already been fired upon, then they will receive a message that it is an invalid shot and be instructed to fire once more. This same logic is used for the computer, except a message will print to the screen informing the player if a shot was a hit or a miss. Each level of difficulty has a different decision structure for shooting for the computer. The easy level of difficulty is completely random, even if there is a hit. Any difficulties past easy incorporates a decision structure that has the computer shoot randomly around the 4 spaces of a hit. On medium, every 8th shot will always be a hit. On medium every 5th, and on impossible every 3rd shot. If the computer’s ships are all sunk, then the player will receive a a message that they have won; however, if all of the player’s ships are sunk they will receive a message that they have lost. Also the player will be asked if they would like to play again.

Our project is entirely constructed using functions. Everything from the 2 game windows, placing ships, and AI are all constructed and drawn on windows using these functions. Our first two functions ask the player if they want to play and what level of difficulty of computer they would like to play against. We then draw a board and then run 5 different place functions. Each place function takes input from the player and draws the ships of different l on the board, but also adds the tiles where the ships are placed to a list, which is called hitListPlayer. It also adds that part of the list to shipListComp to keep tract of ships. The computer then places its ships through 5 similar functions except randomly, and adds the tiles with ships on them to a list called hitspots. We then have a playerShot function that allows the player to shoot the computer’s board, and prints to the player when they have a miss, a hit, or an invalid move. It does this by seeing if that selected tile is within the hitspots or hitlistPlayer list, and then removes that spot from that list. The computer’s method of shooting is done through 4 functions similar to the playerShot function. Depending upon the difficulty of the computer described earlier, the computer will shoot randomly, decide where to shoot based on a decision structure, and/or a certain hit. The function shipcheck determines if a ship has fully sunk, and then prints to the screen if a computer’s ship has been sunk. The Getwinner function is used to determine when the game is over. It goes through both hitistPlayer and hitspots at the end of every turn. If the former is empty, then it prints that the computer has won and ends the game, and if the latter is empty, then it prints that the player has won. If neither is empty then the game continues as normal.

Everything at the moment works for our program, we don’t have any errors that we know of and the only thing that we were potentially thinking about adding were either graphics or sound.

Instructions: Anytime there is a input required from the player, instructions are provided by our functions. For example, when deciding to place your ships horizontally or vertically, we provide the two inputs the player could use.