Project 3
In this project, you will implement a variation of the board game “Battleship”.
You may have played this game as a kid (or an adult). In Battleship, 5 ships are placed on a 10 x 10 grid and then you and your opponent take turns calling out a row/column until all the ships on your opponent’s grid are “hit” and “sunk”. The first player to sink all of his/her opponent's ships wins.

Step 1: create a Battleship class that contains the following properties: self, num_of_ships, rows, cols, and targets. Initialize the num_of_ships to 5, rows to 10, cols to 10, and targets to an empty array.

Step 2: create a method to create a game board called drawBoard. This method will return the BOARD for the user to see an updated board.

Step 3: create a method to set up the game board called setupBoard. This method will create an array of arrays called arr_content. It will be made by multiplying the row * times column. Each array will be initialized with a “.” inside to represent a field. Then, if the number of targets is not 5 (there are no ships on the board yet, or coordinates for the target ships have been repeated), append a random column, and random row to the targets variable. This will contain a “S” to mark the ships on the board. This method will return the board.

Step 4: create a method to check if the user has hit or missed the ships, called checkHitOrMiss. It will take the row and column the user inputted. If the user guessed a column and row with a ship (“S”), It will mark that row with a “X” and return “HIT”. Else if the user guessed a column and row that is empty (“.”), It will mark that row with a “O” and return “MISS”.

Step 5: create a method to check if the user has sunk all the ships. Create a counter variable and set it to 1. For each subarray in the board, if there’s an “X” in the subarray, add 1 to the counter. Then, if the counter equals 5, return true. Otherwise, return false. This will mean that if the user has sunk all ships, the game will be over.

Step 7: import the battleship class on top of the file and create a variable named battleship. Then, create a variable named board that will take in the method from the battle to set up the board. Create a main func in another file. This will initialize the program. Print out the logo of the game from the art file. Print the directions of the game. Then set a variable to equal start_game. This variable will be equal to a prompt asking the user to type in “y” if they want to start the game. If the user input equals “y”, it will run the game_process method. If not, it will print goodbye.

Step 8: create a func to initialize the game process. This function will be called game_process. It will initialize the game by calling the drawBoard method from the battleship class. It will pass in the board variable. While the length of the battleship targets is more than 0, it will ask the user for input and set that input in a variable called inputs. then , it will call the validate_inputs function to validate them. Then, it will call the isGameOver functions, pass the board variable, and check if the game is over, then it will print “Game over, you won!”. If false, it will print “game is not over yet”.

Step 9: the function validate inputs will take the col and row the user inputted. If the values are numeric, it will call the checkHitOrMiss method in the class and pass in the col and row the user inputted along with the board. Finally, it will call the drawBoard method and pass in the board variable.

Step 10: the ask_input function will declare two variables, userow with the inputs from the col and row the user inputted.

Step 11: Call the main func to initialize the program. Pass in the battleship variable from the top of the file.