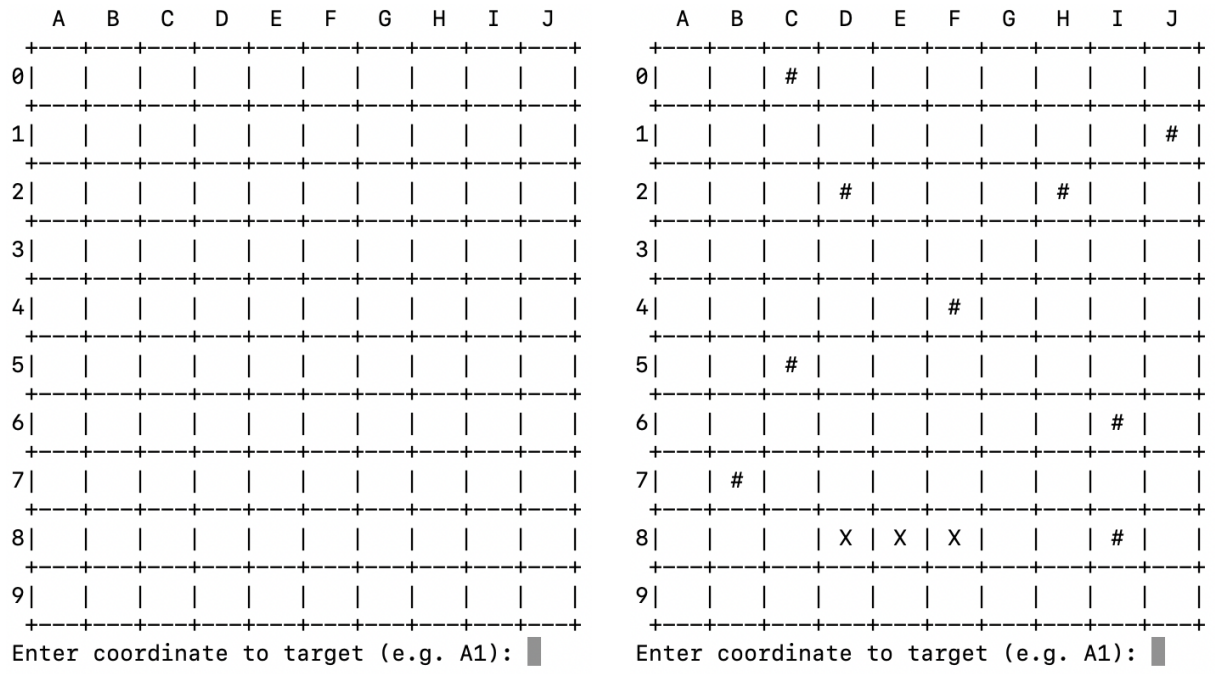
3 Battleship Game: Overview

Battleship (also Battleships or Sea Battle) is a two-player guessing game. It is known worldwide as a pencil and paper game which dates from World War I. It was published by various companies as a pad-and-pencil game in the 1930s, and was released as a plastic board game by Milton Bradley in 1967. The Battleship game is played on a two-dimensional board.

The Board: You are required to consider a square grid board. The individual cells on the grid are identified by letters and numbers as shown in Figure 1.
The Player: In this simplified version of Battleship, you are required to assume only one player.
The Game: At the start of the game, the computer secretly places one (hidden) ship with the size of 4 consecutive cells/blocks on the board in a random location and with a random orientation, either horizontally or vertically. As the location of the ship is concealed from the player, the goal of the player is to "destroy" the ship with the least number of guesses. Hereby, the player takes turns by entering a coordinate of the target cell. On a hit, the cell is marked by an 'X', otherwise by a '#' for a miss. When all cells of the ship are guessed by the player, the ship is sunk and the game is won.
Example: Figure 1 shows an example of the Battleship game on a 10x10 grid.



a) **b)**

Figure 1: a) Example output after the start of the game and **b)** after multiple turns with 3 hits at D8, E8, F8

4 Battleship Game: Implementation

The implementation of this game mainly consists of two phases: Initialization, and Game Phase. In this section, you will find the full details of these two phases.

Initialization Phase:

- At the start of the game, an (empty) board is created, where columns are identified by a letter and rows by a number (see Figure 1 a)).
- Your program then randomly places a ship on the board, where the location of the ship is described by a **random row, column, and orientation**.
- As the ship occupies a number of consecutive cells on the board, the random row and column is used as the first cell of the ship and then extends either vertically (down) or horizontally (right) based on the length of the ship.
- In order to randomly select a row, column, and orientation of the ship, use the randint() function of the random module (as discussed in class). When drawing random numbers for the first cell of the ship, make sure that:
    a. The row and column is within the board dimensions
    b. The ship can actually be placed completely within the board dimensions, i.e. no cell of the ship exceeds the board dimensions.
- As the location of the ship is concealed from the user, you have to find a solution to store the cells that are occupied by the ship without revealing them to the player. For example, use another board that holds the locations of the ship which is not shown to the user and only used to verify a hit or miss.
- After creating the board, the board should be printed on the screen so that the user gets an idea on how the board looks.
- The board dimensions and the size of the ship should be declared as constant (global) variables and the program is expected to dynamically create a board based on the values of these variables (do not "hardcode" board dimensions, size of the ship, etc). This allows you to easily change these values without modifying multiple lines of code. To keep the game logic simple, we restrict our implementation to the following:
    a. The minimum board dimension is 4x4 and the maximum is 10x10. Your code doesn't have to incorporate smaller/larger or non-square dimensions.
    b. The ship length is between 2 and 4 cells.
- The code snippet below shows an example of the first lines of code:

```
1 import random     # imports the random module
2
3 SHIP_SIZE = 4     # constant variable for ship size
4 DIMENSION = 10    # constant variable for board size (square)
```

Game Phase:

After creating the board and placing the ship, your game-phase implementation should consider the following:

Valid/Invalid User Input:

- The player is asked to make a guess by entering a coordinate (location) within the board dimensions. The coordinate consists of the column (a letter) followed by the row (a digit).
- Make sure that the input clearly describes the format of the expected input. For example, valid coordinates are B4, C1, A6, F0, etc. Please note that the column letter is case-sensitive.
- In case the player enters an invalid input or an invalid coordinate, the program informs the player about this and asks for a new coordinate. Examples for invalid input are AB, 1A, A 6, 32, A;5, B,4, etc. or coordinates out of the board dimensions.

Processing Valid Input:

- After a valid user input, the program places the guess on the board, where a hit is denoted as an 'X' and a miss as a '#'.
- The program must also maintain a score indicating the number of guesses.
- After each turn, the program must check if the game is won, i.e. the entire ship is sunk. If so, the program must inform the user that the game is over, display the score and terminate. Otherwise, the board should be printed reflecting the updated cells and ask for another input.
- Before printing the updated board, the screen should be cleared by using the following command (add line 2 and 6 to your code at the respective locations):

```
1 import random
2 import os
3
4 # your code comes here
5
6 os.system("clear")  # clears the screen
```