OnSite Assignment - Battleship

Total Duration: 90 mins

- Code can be written in any object oriented or functional language.
- Code should be idiomatic for the language chosen.
- Unit tests are highly appreciated.
- Use Git (or other version control tools) to make frequent commits.
- Please follow the directory structure and file name conventions of the language you use.
- Follow good coding practices and principles (These carry weightage in the review)
- You have 90 minutes to complete the solution.
    - Remember we are not only looking at completing the solution **but also the approach one takes at solving it**.
    - We will intermittently pair with you to check the solution.

Interview Format

- After receiving the problem statement, you are expected to read it thoroughly.
- For the first 10 minutes, you can ask the interviewer to resolve any doubts and answer questions that you may have.
- You can start working on the problem statement on your own for the next 60 minutes
- Once the 60 minutes are over, the interviewer will pair with you for the remaining 20 minutes.
- If you have already finished the problem statement, the interviewer may ask you to implement additional requirements.

Rules of the Game

- Battleship is a game played between 2 players.
    - Each player will be initialised with an `M * M` grid with 'S' number of ships placed at specified positions on the grid.
- One battleship occupies a single position on the grid.
- It takes a single shot to bring down the battleship of an opponent.
- The objective of the game is to destroy the opponent's battleships. Each player will be given 'T' number of missiles to bring down the fleet of opposition. The player who does the most damage/brings down the ships of the opposition wins
- This will be a simulation of an actual battle wherein each player's actions are given as an input to the program.
- Based on hits / misses of a missile on the opponent, either of the players might be victorious or the game might end as a draw

INPUT

The input for the game will be read from a file which contains the following (numbers below represent line numbers in input file)

1. Contains the size of the battleground `‘M’`
    - `(0 < M < 10 )`
2. Contains the number of ships ’S’ which can be placed on the `M*M` grid
    - `( 0 < S <= M^2/2)`
3. Player 1 ship positions in the grid, position represented by `x1,y1:x2,y2` and so on
4. Player 2 ship positions, they are placed in the grid with the same format as above
5. Tells the total number of missiles players have
    - `(0 < T < 100)`
6. Player 1 missile positions, represented by x1,y1:x2,y2 and so on
7. Player 2 missile positions, represented with the same format as above

A Player's moves will be a colon separated list of target positions (x, y) of missile attacks on the enemy grid.

Note: All the missiles will be shot at the same time for both the players and corresponding ships will be destroyed at the same time as well.

Inputs

1. M is the grid size [Matrix of `M*M`]
2. S is the total ships
3. P1 Ship Positions: `1,1:2,0:2,3:3,4,...` (`x,y` pairs separated by colon)
4. P2 Ship Positions: `0,1:2,0:2,3:3,4,...`
5. T: Total Missiles
6. P1 missile positions: `1,1:2,0`… (`x,y` pairs of length ’T’)
7. P2 missile positions: `1,2:3,0`… (`x,y` pairs of length ‘T’)

Sample Input File

```
5
5
1,1:2,0:2,3:3,4:4,3
0,1:2,3:3,0:3,4:4,1
5
0,1:4,3:2,3:3,1:4,1
0,1:0,0:1,1:2,3:4,3
```

OUTPUT

Output should be written to a file that should contain the following information:

- Player 1 and Player 2 grids after the battleship simulation.
- Alive Battleships denoted with “**B**”
- Dead Battleships with “**X**” (if missile hit the battleship) HIT
- Missile Missed Locations “**O**” (if the missile location didn’t have a ship) MISS
- Final result:
    - P1:total hits
    - P2:total hits
- Game Result

Game Result

- "*Player 1 wins*”, if player 1 does most damage to player 2’s ships
- "*Player 2 wins*”, if player 2 does most damage to player 1’s ships
- "*It is a draw*”, if both players inflict the same damage

Sample Output File

```
Player1
O O _ _ _
_ X _ _ _
B _ _ X _
_ _ _ _ B
_ _ _ X _

Player2
_ X _ _ _
_ _ _ _ _
_ _ _ X _
B O _ _ B
_ X _ O _

P1:3
P2:3
It is a draw
```