Assignment:
We'd like to understand your approach and thought process and for that we have a problem statement that we'd like you to work on.

Please submit your assignment before 10th September 9:00pm HERE.

Key Expectations: Few things that will be important for us and will help us evaluate your submission would be

- Modular and robust code
- Code structuring, readability and adaptability
- Levels completed
- Handling the specified exceptions and edge cases gracefully
- An Object Oriented approach would be greatly appreciated

Guidelines:

1. Create a public repository on your GitHub
2. Upload your project files
3. **Submit your github submission link in the form mentioned (Google Form - Mandatory)** after completing your assignment. Also you can explain your approach by adding points in the READ.ME file (optional).
4. If you have a profile on github or any other platform please share with us in the mail

Note: If you don't have a GitHub account then please upload your project files on your google drive and share the link in the same email and submit in the google form.

There are no assets/resources provided for this assignment. You can use any free assets and/or resources available publicly for use.

Problem Statement

Objective: Build a turn based chess like game, a program that accepts commands to make moves over the cli (command line interface)

Game rules:

- There are 2 *players* playing it
- The game is played on a square *grid* of 5 by 5 blocks
- Each *player* has a team of 5 *characters* that start from their end

- Move commands*:*
  L - left move
  R - right move
  F - front move
  B - back move

  These moves are relative to the *player*.

- *Characters:*
  Pawn: In one move, it moves 1 block straight in any direction (L, R, F, B)
  More characters maybe added subsequently

- The opponent player's character dies, i.e removed from the grid when a player's character is moved to the opponent character's position

Game flow:

- Each player can arrange their characters on their end in any order. They will have 5 pawns to start with and will need to deploy all of them at the start of the game.

- The input for initial character positions will be taken as a list of character names, placing them from left to right on the starting lanes. Once this process is complete for both the players, they can start making moves.

- The application then takes move inputs from the players, alternating between the players, updating the game state in the process.

- Input move format: <character_name>:<move> (e.g. P1:L, P2:R, P3:F, P3:B)

- An invalid input move should be handled and prompted to the user(Invalid input format) and make the player retry their turn.

  Invalid moves for a player:
  a. Input command on a character that does not exist
  b. Character going out of grid bounds.
  c. Invalid move presented for a given character.
  d. Targeting a friendly character, i.e a character from our own team.

- After every turn a 5 by 5 grid is displayed populated by characters.

- Prefix *character* names with *player* names upon displaying the grid. (Player A's characters should be prefixed by A-, ex:- A-P1, A-P3)

- The game ends when one *player* (winner) manages to kill all *characters* of their opponent. The winner is prompted. The players can then choose to play another game.

Example cli flow:

The application prompts for user input. Player 1 selects their characters.

Player1 Input: P3, P2, P5, P4, P1
Current Grid:

-	-	-	-	-
-	-	-	-	-
-	-	-	-	-
-	-	-	-	-
A-P3	A-P2	A-P5	A-P4	A-P1

The characters are valid and can be placed on the grid, the resulting grid is displayed.

Player 2 is asked to select and place their characters on the grid.

Player2 Input: P2, P1, P3, P5, P4
Current Grid:

B-P2	B-P1	B-P3	B-P5	B-P4
-	-	-	-	-
-	-	-	-	-
-	-	-	-	-
A-P3	A-P2	A-P5	A-P4	A-P1

The players can now start making turn based moves, the application halting for user info. Player A chooses to move P1 forward. The grid stays up to date with the current game state.

Player A's Move: P1:F
Current Grid:

B-P2	B-P1	B-P3	B-P5	B-P4
-	-	-	-	-
-	-	-	-	-
-	-	-	-	A-P1
A-P3	A-P2	A-P5	A-P4	-

Player A's Move: P4:F
Current Grid:

B-P2	B-P1	B-P3	B-P5	-
-	-	-	-	B-P4
-	-	-	-	-
-	-	-	-	A-P1
A-P3	A-P2	A-P5	A-P4	-

Player B's Move: P1:F
Current Grid:

B-P2	B-P1	B-P3	B-P5	-
-	-	-	-	B-P4
-	-	-	-	A-P1
-	-	-	-	-
A-P3	A-P2	A-P5	A-P4	-

Player B's Move: P4:F
Current Grid:

B-P2	B-P1	B-P3	B-P5	-
-	-	-	-	-
-	-	-	-	B-P4
-	-	-	-	-
A-P3	A-P2	A-P5	A-P4	-

This move registers a kill and removes 'A-P1' from the grid, it's place gets taken by 'B-P4'.

Implementing all the requirements specified above makes you eligible for level 2:

Level 2:

Add the following character types to the game:

1. Hero1: In one move, it moves 2 blocks straight in any direction, and **kills anything in its path**
2. Hero2: In one move, it moves 2 blocks diagonally in any direction, and **kills anything in its path**

H1:L means Hero1 to left, H2:FL means move H2 to forward-left direction

Implement all R,L,F,B for H1 and FL, FR, BL, BR for H2

Level 3:
Add another character type:

Hero 3: In one move, it moves 2 steps straight and one to the side, and kills only where it finally lands

H3:FR means it moves 2 steps front and one to the right,
H3:RF means it moves 2 steps right and one to the front

Implement all FL, FR, BL, BR, RF, RB, LF, and LB for H3

Please submit your assignment before 10th September 9:00pm HERE.

Good Luck!
Feel free to reach out to us on recruitment@hitwicket.com if you have any queries

Look forward to moving to the next round with you!