class LudoGame:

The LudoGame object represents the game as played. The class should contain information about the players and information about the board.

Data members:

Must be private

Amount of data members and kind of data members are programmer's choice

Required Methods:

play_game (self, param_1, param_2)

- Parameter 1: Players list
  List of positions players choose, like **['A', 'C']** means two players will play the game at position A and C.
  Sample argument: players = ['A', 'B']

- Parameter 2: Turns list
  List of tuples with each tuple a roll for one player. For example, [('A', 6), ('A', 4), ('C', 5)] means player A rolls 6, then rolls 4, and player C rolls 5.
  Sample argument: turns = [('A', 6),('A', 1),('B', 6),('B', 2),('C', 6),('C', 3),('D', 6),('D', 4)]

1. Create player objects using the players list that is pass in as a parameter
2. Move the tokens according to the turns list following the Move Priority (see next page)
3. Update the tokens position
4. Update the player's game state (whether finished the game or not).
5. **Return** a list of strings representing the current spaces of all tokens for each player in the list after moving the tokens following the rules described above.
   - 'H' for Home Yard
   - 'R' for Ready To Go Position
   - 'E' for finished position
   - Other letters/numbers for the space the token has landed on.

get_player_by_position (self, param_1)

- Parameter 1: player's position as a string
  Sample method call: player_A = game.get_player_by_position('A')

1. **Return** the player object.
   - If parameter is an invalid string, **Return** "Player not found!"

class LudoGame (cont'd):

Required Methods:

move_token (self, param_1, param_2, param_3)

- Parameter 1: Player object
- Parameter 2: Token name ('p' or 'q') **(type: string)**
- Parameter 3: Steps the token will move on the board **(type: int)**.

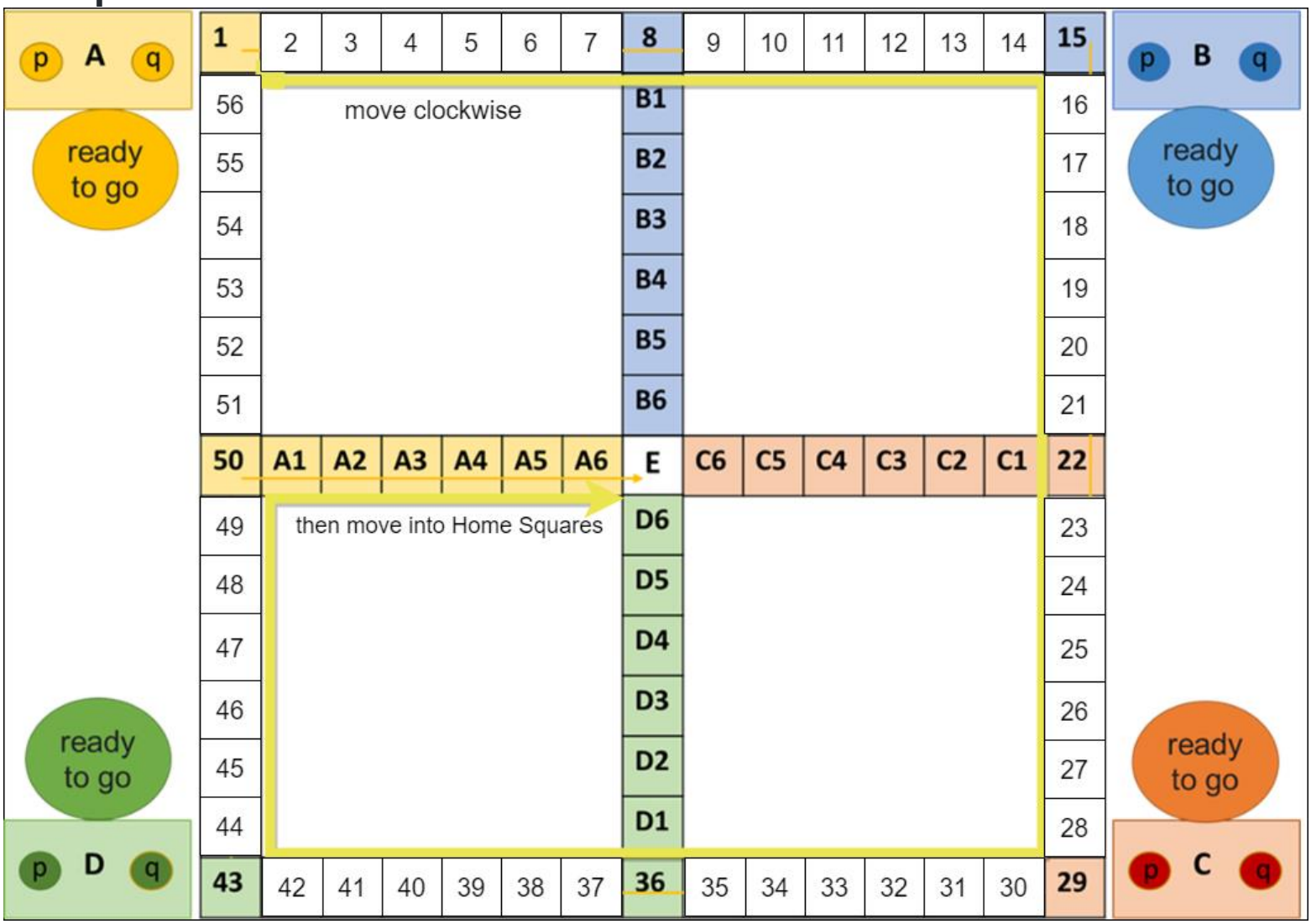
1. Take care of one token moving on the board.
2. Update the token's total steps
3. Take care of kicking out other opponent tokens as needed.
4. **The play_game method will use this method**.

Move Priority

If the player can make a valid move for each of their tokens, the player' move priority is as follows:

1. Rolled a 6
   - Moving from your 'H' to 'R' is first priority (if you still have tokens in your Home Yard).
   - If both tokens are in 'H', token 'p' is moved out first.
2. If token is in the home square and die roll is exactly what is needed to reach the final space, move token to final space.
3. If token can move and land on opponent's token(s), then land on opponent's token.
4. Furthest token away from final space moves

Example Board:



Home Yard ('H')	Where your p and q tokens are initially placed in the board.
Ready to go ('R')	When a player has tokens left in their Home Yard and rolls a 6, they can move a token to their own Ready to Go space in the board.
Home Squares	The six steps on the board denoted as [Player position][1 – 6] such as A1, A2, A3, A4, A5, A6. Tokens in on these spaces cannot be kicked by another opponent.
Final Square ('E')	The square denoted as "E" in the middle of board. Tokens can only move to this square with an **exact** roll.
Position A	Starts on space 1 and space 50 is the last space number before A's Home Squares.
Position B	Starts on space 15 and space 8 is the last space number before B's Home Squares.
Position C	Starts on space 29 and space 22 is the last space number before C's Home Squares.
Position D	Starts on space 43 and space 36 is the last space number before D's Home Squares.

class Player:

The Player object represents the player who plays the game at a certain position.

Required Information (may have more):

a. The position the player choose (A, B, C or D)

b. Start and End space for this player

c. Current position of the player's two tokens:
Note: current position will be represented as **strings**, for methods get_space_name AND play_game.

- 'H' (Home Yard)
- 'R' (First Space after Home Yard)
- 'E' (Final Space)
- A space on your board which can be a number or letter and number. For example: '20' or 'C4'

d. The current state of the player: Won and finished? Or is still playing?

Data members:

Data members must be private.
Programmer's responsibility to store the required information above into their own data members.
Programmers are encouraged to code their own data members.

Required Methods:

get_completed (self)

- Takes no parameters

1. **Return** True or False if the player has finished or not finished the game

get_token_p_step_count (self)

- Takes no parameters

1. **Returns** the total steps the token p has taken on the board **(int)**

get_token_q_step_count (self)

- Takes no parameters

1. **Returns** the total steps the token q has taken on the board **(int)**

Pertains to both *get_token_p_step_count and get_token_q_step_count*

'H' position, use steps = -1

'R' position, use steps = 0

The total step should not be larger than 57.

Please note that if the token is bounced back in the home squares, this bounced part should be subtracted from the step count.

For example, when token p is at space A5, the total step is 55 now. If it moves 5 steps and bounces back to A4, the total step should be 54, not 60.

class Player (cont'd):

get_space_name (param_1)

- Parameter 1: total steps of the token

1. **Returns** the name of the space the token has landed on the board as a **string**.
    - If token is at 'H', **return** 'H'
    - If token is at 'R', **return** 'R'
    - If token is at 'E', **return** 'E'
    - Home Squares (must be denoted as [player letter][step 1 – 6]
        - If the Player is position A, and the argument passed in is 51, **return** "A1"
        - If the Player is position B, and the argument passed is 56, **return** "B6"
    - If token is on a normal space, return space number as string.
        - If the Player is position A, and the argument passed in is 20, **return** "20"
        - If the Player is position B, and the argument passed is 20, **return** "34"
        - If the Player is position C, and the argument passed is 20, **return** "48"
        - If the Player is position D, and the argument passed is 20, **return** "6"
            - *Note that Player B starts at Space 15.*
            - *Note that Player C starts at Space 29.*
            - *Note that Player B starts at Space 43.*

General Rules:

- 2 – 4 Players
- Each player has 2 tokens.
- Both tokens start in player's 'H'.

Player Turn Rules:

- Player rolls one die.
- A player token is consider "in play" when it has successfully moved out of the 'H'.
  - Must roll a **6** to move token out of their 'H' onto 'R'
  - If player fails to roll a 6 while in their 'H' AND they have no tokens "in play", player turn ends

"in play" token rules

- For "in play" tokens, the die roll indicates the number of spaces they can move along their track.
  - Rolling a 6 earns the player an additional roll in that turn.
    - ❖ If the bonus roll results in a 6 again, there is no additional roll.
  - Move spaces in clockwise fashion.

Landing on opponent space

- If your token moves on a space already occupied by an opponent, *the opponent token will be returned to their 'H'*

Landing on your own token space

- If your token moves on a space already occupied by your other player token, the two tokens **will stack** and they can now be moved as one token until they reach their finishing square.
  - ❖ If an opponent lands on a space, already occupied by your stacked tokens, **both** tokens will be returned to your 'H' **unstacked**.
  - ❖ You cannot stack your tokens on your 'R'.

- After **clockwise 50 steps** (without being returned back to 'H'), a token will reach their Home Squares.
  - No opponents may enter this space.
  - Home Squares consist of 6 spaces such as A1, A2, A3, A4, A5, A6. (Or steps 51-56).
  - A player can only land on 'E' (step 57) on an **exact roll**.
    - For not exact rolls, player moves away from step 57 for the remainder of their die roll.

Win Condition: Player to get both of their tokens on 'E' will win the game. Game continues until there is only one player left.

Readme Test Code:

For testing purposes, **we will not use the random function to generate the dice number for each roll**.

Instead, we will pass in the roll numbers manually when we call the method. *The list of tuples*

We also don't require a GUI for this project and **all the output will be in the text format**.

You can improve your code later on after you finish the required part to make it your own portfolio project.

```
players = ['A', 'B']
turns = [('A', 6), ('A', 4), ('A', 5), ('A', 4), ('B', 6), ('B', 4), ('B', 1), ('B', 2),
('A', 6), ('A', 4), ('A', 6), ('A', 3), ('A', 5), ('A', 1), ('A', 5), ('A', 4)]

game = LudoGame()
current_tokens_space = game.play_game(players, turns)
player_A = game.get_player_by_position('A')
print(player_A.get_completed())
print(player_A.get_token_p_step_count())
print(current_tokens_space)
player_B = game.get_player_by_position('B')
print(player_B.get_space_name(55))
```

The output will be:

```
False
28
['28', '28', '21', 'H']
B5
```