Snakes & Ladders

Problem Statement

You have to design & implement a **Snakes and Ladders** game that supports the following functionality:

Requirements:

Mandatory Requirements:

You have to take a configuration (can be a yml/json file) with the following parameters.

- Number of players: N
- Board Size: BS (BS x BS)
- Number of Snakes: S
- Number of Ladders: L
- Number of Dies: D
- Movement Strategy: MS

Note: Movement strategy is either SUM (sum of numbers on dies), MAX (max of numbers on dies), MIN (min of number on dies).
You will be given a sample input to populate the board. Post which the game has to be simulated among N players.

Rules:

- Snake always takes you to the cell where its tail is, and has to be a number less than where you are at currently.
- Ladder takes you up (strictly).
- If a player (A) comes to a cell where another player (B) is placed already, the previously placed player (B) has to start again from 1.

Optional Extensions:

- Using the configuration you have to generate a random valid board & **devise proper rules** for placing objects on the board.
- Write unit tests to validate all implemented functionality and their edge cases.
- Addition of special objects:
    - **Crocodile**, which takes you exactly 5 steps back.
    - **Mine** which holds you for 2 turns.

Expectations:

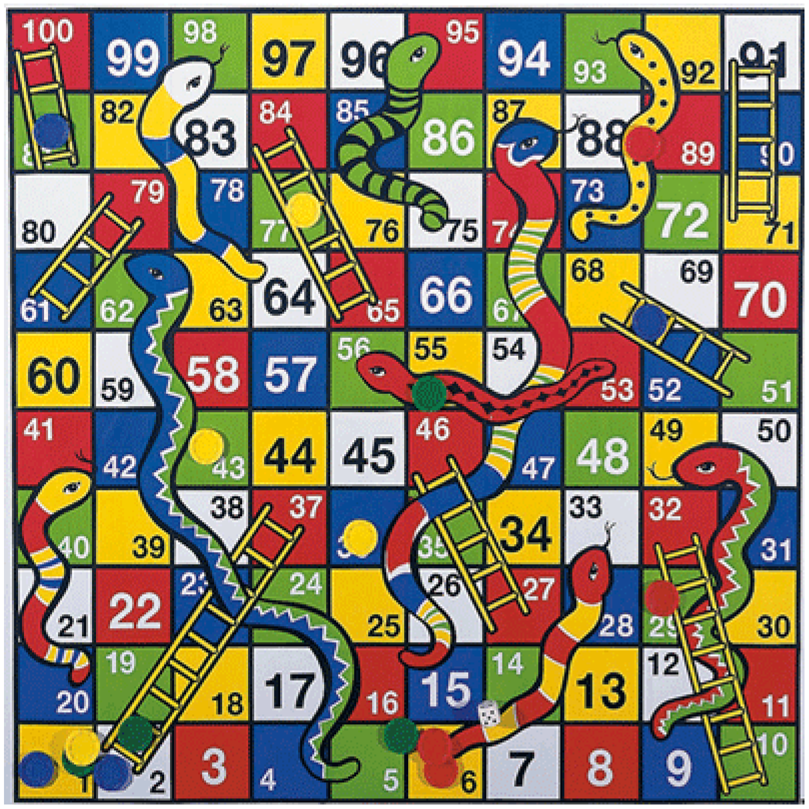
- Your code should cover all the functionality explained under “Mandatory Requirements”.
- You should implement at least one of “Optional Extensions”, preferably all.

- Implement a driver program which drives all above rules, and gives turns to players in a round robin fashion to roll dies.
- Proper logging of all events, e.g rolling dies, movement to new cell, encountering snakes/ladders etc.
- Code has to handle all edge cases
- Code should be modular enough to add further extensions or rules, without much changes.
- A manual override must exist for the interviewer to verify the edge cases/ to write unit tests. The program should take the following as input:
    - Starting location of each player.
    - The D die values that each player rolled in a turn.

Guidelines:

- The die roll can be implemented using a random function.

Sample Input and Output:



Input format:

- Total Snakes S
- Following S lines contains pair (Snake's Head and Snake's Tail)

- Total Ladders L
- Following L lines contains pair (Ladder bottom and Ladder top)
- N no of players
- Following N lines contains names & starting locations of each Player
- An override to manually enter the D die values that each player rolled in each turn. (Absent in example. Any input format is fine)

```
9
62 5
33 6
49 9
88 16
41 20
56 53
98 64
93 73
95 75
8
2 37
27 46
10 32
51 68
61 79
65 84
71 91
81 100
2
Gaurav 1
Sagar 1
```

Output:

```
Gaurav rolled a 6 and moved from 0 to 6
Sagar rolled a 1 and moved from 0 to 1
Gaurav rolled a 6 and moved from 6 to 12
Sagar rolled a 4 and moved from 1 to 5
Gaurav rolled a 4 and moved from 12 to 16
Sagar rolled a 6 and moved from 5 to 11
Gaurav rolled a 5 and moved from 16 to 21
Sagar rolled a 4 and moved from 11 to 15
Gaurav rolled a 1 and moved from 21 to 22
```

```
Sagar rolled a 6 and moved from 15 to 21
Gaurav rolled a 6 and moved from 22 to 28
Sagar rolled a 2 and moved from 21 to 23
Gaurav rolled a 6 and moved from 28 to 34
Sagar rolled a 6 and moved from 23 to 29
Gaurav rolled a 5 and moved from 34 to 39
Sagar rolled a 2 and moved from 29 to 31
Gaurav rolled a 2 and bitten by snake at 41 and moved from 41 to 20
Sagar rolled a 5 and moved from 31 to 36
Gaurav rolled a 3 and moved from 20 to 23
Sagar rolled a 5 and bitten by snake at 41 and moved from 41 to 20
Gaurav rolled a 6 and moved from 23 to 29
Sagar rolled a 3 and moved from 20 to 23
Gaurav rolled a 2 and moved from 29 to 31
Sagar rolled a 3 and moved from 23 to 26
Gaurav rolled a 3 and moved from 31 to 34
Sagar rolled a 5 and moved from 26 to 31
Gaurav rolled a 3 and moved from 34 to 37
Sagar rolled a 4 and moved from 31 to 35
Gaurav rolled a 2 and moved from 37 to 39
Sagar rolled a 5 and moved from 35 to 40
Gaurav rolled a 2 and bitten by snake at 41 and moved from 41 to 20
Sagar rolled a 5 and moved from 40 to 45
Gaurav rolled a 2 and moved from 20 to 22
Sagar rolled a 6 and climbed the ladder at 51 moved from 51 to 68
Gaurav rolled a 3 and moved from 22 to 25
Sagar rolled a 3 and climbed the ladder at 71 and moved from 71 to 91
Gaurav rolled a 5 and moved from 25 to 30
Sagar rolled a 2 and bitten by snake at 93 and moved from 93 to 73
Gaurav rolled a 5 and moved from 30 to 35
Sagar rolled a 6 and moved from 73 to 79
Gaurav rolled a 5 and moved from 35 to 40
Sagar rolled a 1 and moved from 79 to 80
Gaurav rolled a 4 and moved from 40 to 44
Sagar rolled a 2 and moved from 80 to 82
Gaurav rolled a 5 and bitten by snake at 49 and moved from 49 to 9
Sagar rolled a 4 and moved from 82 to 86
Gaurav rolled a 1 and climbed the ladder at 10 and moved from 10 to 32
Sagar rolled a 6 and moved from 86 to 92
Gaurav rolled a 3 and moved from 32 to 35
```

```
Sagar rolled a 4 and moved from 92 to 96
Gaurav rolled a 1 and moved from 35 to 36
Sagar rolled a 1 and moved from 96 to 97
Gaurav rolled a 1 and moved from 36 to 37
Sagar rolled a 5 and moved from 97 to 97
Gaurav rolled a 6 and moved from 36 to 42

Sagar rolled a 3 and moved from 97 to 100
```

How will you be evaluated

- Functional coverage
- Testability
- Application of OO design principles
- Code modularity, readability
- Separation of concerns