Introduction to Programming - CSE101
Assignment 4

GridWorld Game

You have to create a game in python where the **player** has to reach the goal position from the start position, collecting all the randomly spawned **rewards** and avoiding **obstacles**.

- The grid is of size n X n. You have to take n as an input from the user. The start and goal positions should be randomly placed on the boundary.
- There are obstacles in some of the cells of the grid and their positions should also be random.
- The player can make moves like up, down, left and right and 2 special moves - rotate clockwise and rotate anticlockwise. You have to implement functions for each of the moves to achieve their functionality. The moves are case insensitive.
- A move is to be taken as input from the player. For instance, R4D3L2U1 means move 4 units to the right, then 3 units down, then 2 units to the left and finally 1 unit upwards. (The input need not necessarily contain all R, L, U and D. For example, R4D3 is also a valid input). You have to take these inputs from the player until it loses all its energy or reaches the goal position.
- The special moves (rotate anti/clockwise) must also be taken as input from the player. If the input is A3, it means that you have to rotate the grid anti-clockwise 3 times. Similarly, C3 means 3 times clockwise rotation. All rotations are by 90^{o}. Note that when you rotate the grid, the coordinates of the player and goal do not change; only the grid cells get rotated. Every rotation command reduces the player's energy by n//3 units.
- The score at the start is basically the energy of the player. The initial value of the energy is 2n units. Every time, the player consumes food, the player's energy increases by value times n units where each food has its own value and every time it hits an obstacle it loses 4n units.
- Also, for every move, it loses 1 unit of energy(Energy lost for R4D3L2U1 is 4+3+2+1 = 10 units).
- If the player is at the boundary and it makes a move which can take it out of the boundary, then the player should appear at the opposite side of the grid. For instance, if it is at the left boundary and it moves left, so it should appear at the right side of the grid on the same row.

How should the game look?

You have to show the following details after every move:

- The energy level of the player.
- The state of the grid.

When you execute a command, say R4D3L2U1, your console should look something like : https://drive.google.com/file/d/1FgIqU8I3FgLbv7bdXeiAeqegI4k8AQH0/view?usp=sharing
This sample doesn't have any obstacles or rewards, you have to spawn them yourself.

You must make the following classes:

1. **Grid** : It will have the following data members
    a. N : size of the grid
    b. start : original position of the player
    c. goal: final position of the player
    d. myObstacles: an array of obstacles
    e. myRewards: an array of rewards

    It will have the following functions:
    a. rotateClockwise(n) : rotates the grid clockwise n times by 90^o.
    b. rotateAnticlockwise(n) : rotates the grid anti-clockwise n times by 90^o.
    c. showGrid() : prints the grid on the console. Obstacles are represented by '#', rewards by its corresponding value, empty cells by '.' and the player by 'O'

    For rotate functions, if after rotation the player's position will coincide with that of an obstacle, you have to print a message saying that the grid can't be rotated.

2. **Obstacle**: An obstacle on the grid will be represented by a '#' character. It will have the following data members:
    a. x: x coordinate of the obstacle
    b. y: y coordinate of the obstacle
3. **Reward**: A reward can have values between 1 and 9. It will have the following data members:
    a. x: x coordinate of the reward
    b. y: y coordinate of the reward
    c. value: an integer v, $1 \leq v \leq 9$.
4. **Player**: It will have the following data members:
    a. x: the x coordinate of the player
    b. y: the y coordinate of the player
    c. energy: the energy of the player

    It will have the following function:
    a. makeMove(s): where s is to be taken as input from the player and corresponding move is to be made.

Following are the libraries, you can use for this assignment:

1. os
2. time
3. random

You don't have to use pygame or any similar library, everything is to be printed on the console.

Submission Instructions:

1. The filename should be rollno_a4.py. (You have to submit just one file.)
2. Your code should be **well-commented**.
3. You must not change the class and function definitions.
4. Do not use any libraries other than mentioned above.