PIEEEXMan

Time limit: *1000 ms*
Memory limit: *256 MB*

⇄Interactive

Everyone knows about the classic arcade game Pac-Man. This year, you're about to be playing in a special variation of it, called the PIEEEXMan. In this variation PIEEEXMan, our hero, munches his way around the maze collecting cherries, while Bob the Bear, our anti-hero, can't eat our hero, like the monsters in the original game, so instead he is looking to steal as many cherries as he can.

The maze is a 0-based $(2N+1) \times (2M+1)$ matrix of characters A. The cells are represented by the pairs (i, j) where i and j are odd integers. Moving throughout the cells can be done in the 4 directions: upward, downward, leftward and rightward; however it may be restricted by a wall. The character of the matrix directly in the direction you are facing encodes this information. For example, if $A_{i-1,j}$ contains a wall, then you cannot go from (i, j) to $(i-2, j)$ (which is the cell located immediately upward).

You are playing in the role of PIEEXMan, while the judge is playing Bob. You start first.

Interaction

The judge will print two integers N and M, followed by a A, which is encoded as follows:

Character	Encoding	Where
`1`	PIEEXMan's initial position	only one, located in a cell
`2`	Bob the Bear's initial position	only one, located in cell
`#`	Wall	not in a cell
`.`	Empty	anywhere
`@`	Cherry	located in a cell

The moves you can make are as follows:

Character	Encoding
`U`	Move upward

Character	Encoding
`D`	Move downward
`L`	Move leftward
`R`	Move rightward
`W`	Wait

The judge makes the same type of moves. The game ends when the judge will print a move followed by the `X` character. In this case, you must end the interactions to get a proper verdict.

Download materials & maps

For this challenge there are 6 maps and numerous **judges** whith whom you will compete to collect the cherries. There are 6 examples, each of them coresponding to one map and one **judge**. Not all **judges** are used in the examples.

You can download the maps in txt and bmp format. The txt is the same as the one received in the interaction. The image is just a nice graphic representation of the map to help you better visualize it.

Scoring

Let A be the number of cherries you have collected, B be the number of cherries the judge has collected and C be the total number of cherries in the maze, then you score will be $5 \cdot (1 + \frac{A-B-1}{C+1})$.

Constraints and notes

- This task is **NOT** **adaptive**
- $1 \leq N, M \leq 30$
- A cherry may not be collected twice

Simulation

To see a simulation of a game follow the steps below

- Select a map that you would like to see a simulation for
- Open the **moves** link for the selected map
- Enter **all** the moves into the `moves` input box which is located below the simulation panel
- Open the **state** link for the selected map (the state is the same as the one that can be downloaded in the materials section)
- Enter the map description into the `state` input box which is located below the simulation panel
- Click the orange `Reload` button
- To see the simulation click the `play` button

Hints

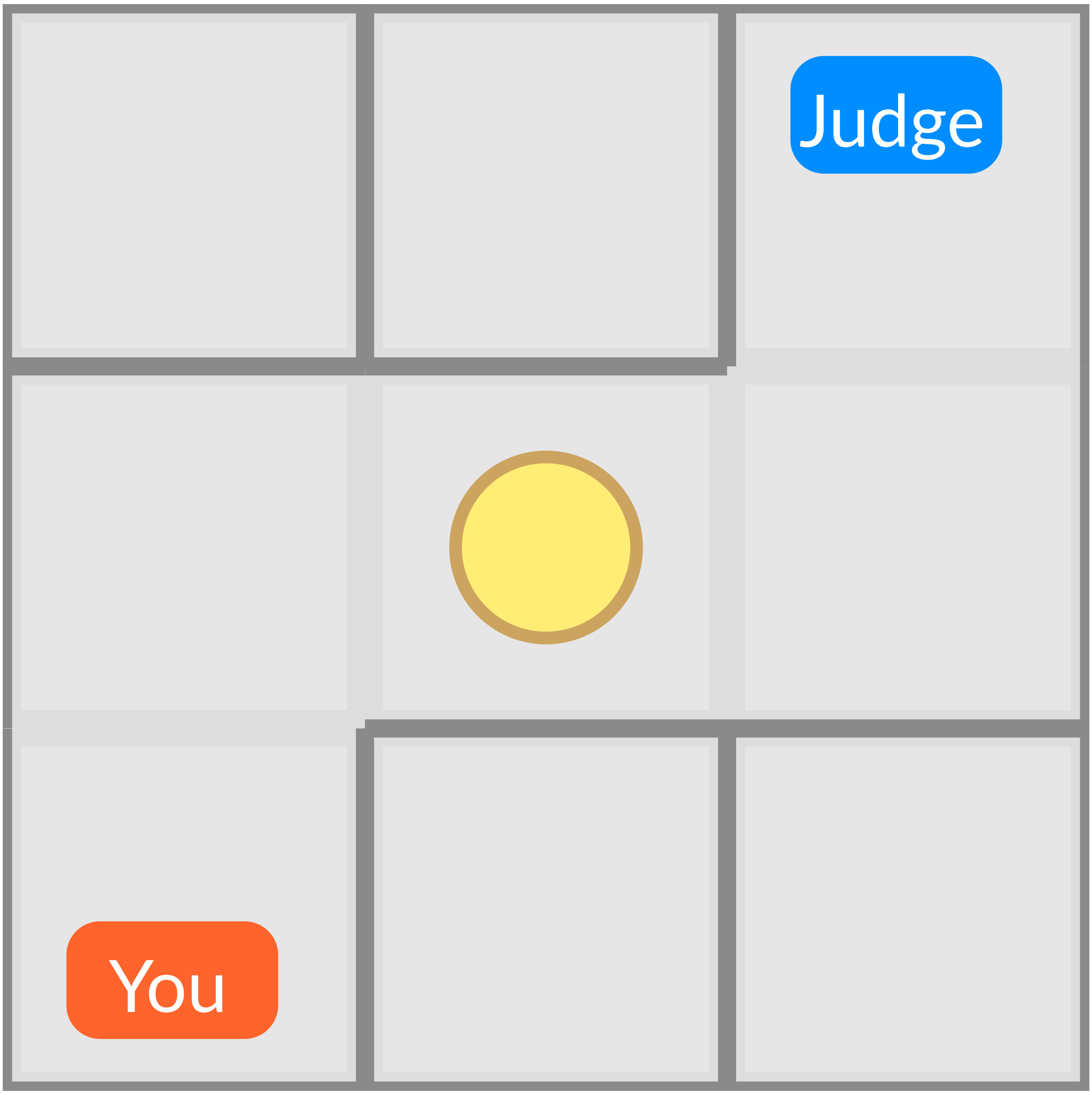
- You can speed up the simulation replay (maximum of $16\ \times$)
- You can enter your own simulation into the `moves` input box. Just beware that it **must** contain both your moves and the judges moves.
- You can use the simulation to see a replay of your code on the examples. To do this, click the `Run Examples` button. Expand an example by clicking the `>` character in the `examples panel`. Carefully

select the `interaction` and paste it into the 2 `input boxes`

- Beware that the `interaction` might look like the following (see below). You must split it accordingly before pressing the `Reload` button.

```
3 3
#######
#.#.#2#
#####.#
#..@..#
#.#####
#1#.#.#
#######UWRWX
```

Map Number	Example of Moves	Map state
0	Moves	State
1	Moves	State
2	Moves	State
3	Moves	State
4	Moves	State
5	Moves	State



You
0

Judge
0

Coins left: 1

‹ › ▶ Move: 0 ▾ of 4 Speed: 1x ▾

Loaded

Moves:

```
U
W
R
W
X
```

State:

```
3 3
#######
#####2#
#####.#
#..@..#
#.#####
#1#####
#######
```