# ICPC PACNW 2016 B - Buggy Robot

**Time limit:** 0.6s **Memory limit:** 256M

You are trying to program a robot to navigate through a 2-dimensional maze and find the exit.

The maze can be represented as a grid with $n$ rows and $m$ columns. Some grid cells have obstacles that the robot cannot pass. The other cells are empty, which the robot can freely pass. Exactly one of the empty cells in the grid is marked as the exit, and the robot will exit the maze immediately once it reaches there.

You can program the robot by sending it a *command string*. A command string consists of characters `L`, `U`, `R`, `D`, corresponding to the directions left, up, right, down, respectively. The robot will then start executing the commands, by moving to an adjacent cell in the directions specified by the command string. If the robot would run into an obstacle or off the edge of the grid, it will ignore the command, but it will continue on to the remaining commands. The robot will also ignore all commands after reaching the exit cell.

Your friend sent you a draft of a command string, but you quickly realize that the command string will not necessarily take the robot to the exit. You would like to fix the string so that the robot will reach the exit square. In one second, you can delete an arbitrary character, or add an arbitrary character at an arbitrary position. Find how quickly you can fix your friend's command string.

You do not care how long it takes the robot to find the exit, but only how long it takes to repair the command string.

## Input

The first line of input contains the two integers $n$ and $m$ ($1 \leq n, m \leq 50$).

Each of the next $n$ lines contains $m$ characters, describing the corresponding row of the grid. Empty cells are denoted as `.`, the robot's initial position is denoted as `R`, obstacles are denoted as `#`, and the exit is denoted as `E`.

The next and final line of input contains your friend's command string, consisting of between $1$ and $50$ characters, inclusive.

It is guaranteed that the grid contains exactly one `R` and one `E`, and that there is always a path from `R` to `E`.

## Output

Print, on a single line, a single integer indicating the minimum amount of time to fix the program.

## Sample Input 1

```
3 3
R..
.#.
..E
LRDD
```

## Sample Output 1

```
1
```

## Sample Input 2

```
2 4
R.#.
#..E
RRUUDDRRUUUU
```

## Sample Output 2

```
0
```