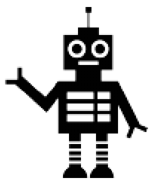
7693 Buggy Robot



Your friend just bought a new programmable robot and has asked for your help.

The robot operates in a 2D grid that may contain obstacles. The environment has a known start location and a known goal location. The robot is controlled with a string consisting of commands '`L`', '`R`', '`U`', and '`D`', which respectively instruct the robot to move one square to the left, right, up or down in the grid. The robot will ignore a command (but continue with future commands) if the command would cause it to leave the grid or to run into an obstacle. If the robot were to reach the goal position, it immediately stops its program with success (even if further commands exist).

Your friend has already programmed the robot with a command string, but the problem is that your friend is not a very good programmer, and so it may be that the commands do not lead the robot successfully to the goal. You would like to fix the string so that the robot will reach the goal, but you do not want your friend to feel bad, so you are hoping to make as few changes to the program as are needed. A single change consists either of deleting an arbitrary character of the command string, or inserting an arbitrary command anywhere within the string.

Input

The input file contains several test cases, each of them as described below.

The first line of the input contains the two integers H and W that respectively define the height and width of the grid such that $1 \leq H, W \leq 50$.

The next H lines each has W characters, describing the corresponding row of the grid, such that the character '`S`' designates the starting location, '`G`' designates the goal location, '`#`' designates an obstacle, and '`.`' designates an empty location. There will always be precisely one '`S`' and one '`G`' in the grid, and there will always exist an unobstructed path from the start position to the goal position.

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

Output

For each test case, output a single integer indicating the minimum number of changes that are needed to fix the program.

Explanations:

- If we consider Sample Input 1, we see that your friend's command string of `DRRDD` does not succeed. The initial `D` moves the robot one spot down. From there, the `R` (and the subsequent `R`) are ignored because of the obstacle to the robot's right. The subsequent `D` moves the robot down once again and the final `D` is ignored. However, by deleting the initial `D`, we can rely on the command string `RRDD` which does successfully lead the robot to the goal.

- If we consider Sample Input 2, we find that your friend's original command string `LDLDLLDR` is flawed. However, if we insert the single command `U` after the fifth command, the resulting string `LDLDLULDR` successfully guides the robot to the goal. It starts by moving left-down-left; the next down command is ignored because the robot is not allowed to leave the grid. The subsequent left-up-left completes the path to the goal (and the final `DR` commands are ignored as the robot stops immediately upon reaching the goal).

- With Sample Input 3, your friend's command string can be corrected with two changes, for example with the new command `ULDLDLLDLR` (although this is not the only way to use two changes to produce a valid sequence).
- In the fourth example, the robot will touch the exit cell during the given path, so we don't need to do anything.
- In the fifth example, we can insert the character '`R`' into the middle to get '`LRRDD`'.

Sample Input

```
3 3
S..
.#.
..G
DRRDD
3 7
.......
.G.#.S.
.......
LDLDLLDR
3 7
.#.....
.G.##S.
.......
LDLDLLDR
2 4
S.#.
#..G
RRUUDDRRUUUU
3 3
R..
.#.
..E
LRDD
```

Sample Output

```
1
1
2
0
1
```