Segfault

Little Petar and his fellow hacker Little Nikolaj enjoy playing the "SIGSEGV" game against one another in their free time, as all hackers do. In this game, two hackers compete for supremacy over a victim's RAM. To achieve this, they need to develop programs that will capture as many memory locations as possible for their own.

The rules of the game are as follows:

- The RAM is represented by a matrix, RAM[][]; each field of this matrix may either be **free**, owned by Petar or Nikolaj, or owned by the **victim**.
- Petar and Nikolaj's programs begin execution from a field of this matrix, and are able to perform only a single type of command: "move to the field directly up, down, left or right from the current field, and attempt capturing it."
- If a program tries to capture a field that is not free and also not owned by its owner, it will receive a **segfault** (*segmentation fault*) signal and **terminate immediately**; however, the hacker owning it will **retain** all the memory that the program captured.
- Petar and Nikolaj may launch **many** programs **simultaneously**, should they have access to a multi-core computer. The programs may be launched **only** at the **start** of the game.

Each hacker has gained an unfair advantage over the other:

- Nikolaj managed to make his capturing scheme faster, so if Petar and Nikolaj both attempt to capture the same (free) field at the same time, Nikolaj's program will capture it and Petar's program will terminate.
- Nikolaj owns a K-core computer, so he may launch K programs at the start of the game.
- Petar found out the entire initial state of the memory, as well as all of Nikolaj's programs. As Nikolaj doesn't know the initial state, he programmed his programs to be very simple: they begin at a given field and will always move in a single given direction (until encountering either the end of the matrix or a non-free field not owned by Nikolaj, after which it terminates).
- Nikolaj's programs will start their execution only after Petar's programs capture the initial field.
- Due to a virus that Nikolaj inserted in Petar's programs, all of them must start from the same field.
- Petar owns the latest Pintel Core i∞ processor that has infinitely many cores; as such, Petar may initially launch an arbitrary number of programs (however, they must all start from the same field, as mentioned before).

Petar is interested in knowing how many fields of the RAM can he capture if he plays optimally.

Input

The first line of the standard input contains two natural numbers N and M separated by a space; these represent the number of rows and columns of the RAM[][] matrix, respectively.

The following N lines contain an M-character string each, representing the initial state of the memory (RAM[i][j] is represented by the j-th character in the i-th line):

- Character '.' represents a free field;
- Character '#' represents a field owned by the victim;
- Character 'S' represents the field from which all of Petar's programs must start.

The following line contains a natural number K, the number of cores of Nikolaj's processor. Each of the following K lines contains the description of one of Nikolaj's programs, in the form X Y DIR, where X and Y are integers representing the program's starting field, and DIR is a character representing the program's only direction of movement ('U' - up, 'D' - down, 'L' - left, 'R' - right).

Output

In the first and only line of the standard output, print the number of memory locations that Petar may capture, assuming he plays optimally.

Example

Input:
6 5
#.S.#
.....
.....
...##
##...
...##
2
2 1 R
3 5 L

Output:
18

Explanation

Petar can achieve the optimal strategy with only launching a single program; this program should make its first two steps downwards, and the state of the RAM will change as follows (let 'P' represent fields owned by Petar, and 'N' represent the fields owned by Nikolaj):

#.P.# #.P.# #.P.#

..... N.P.. NNP..

..... => ....N => ..PNN

...## => ...## => ...##

##... ##... ##...

...## ...## ...##

Both of Nikolaj's programs will hence receive a segfault in the next step, leaving Petar free to capture all of the remaining fields of the matrix (18 in total).

Constraints

- $1 <= N, M <= 1000$
- $1 <= K <= 5 * 10^5$
- $1 <= X <= N$
- $1 <= Y <= M$
- RAM[X][Y] is not equal to '#'.