B. Serial Time!

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

The Cereal Guy's friend Serial Guy likes to watch soap operas. An episode is about to start, and he hasn't washed his plate yet. But he decided to at least put in under the tap to be filled with water. The plate can be represented by a parallelepiped $k \times n \times m$, that is, it has k layers (the first layer is the upper one), each of which is a rectangle $n \times m$ with empty squares ('.') and obstacles ('#'). The water can only be present in the empty squares. The tap is positioned above the square (x, y) of the first layer, it is guaranteed that this square is empty. Every minute
a cubical unit of water falls into the plate. Find out in how many minutes the Serial Guy should unglue himself from the soap opera and turn the water off for it not to overfill the plate. That is, you should find the moment of time when the plate is absolutely full and is going to be overfilled in the next moment.

Note: the water fills all the area within reach (see sample 4). Water flows in **each** of the 6 directions, through faces of $1 \times 1 \times 1$ cubes.

Input

The first line contains three numbers k, n, m $(1 \le k, n, m \le 10)$ which are the sizes of the plate. Then follow k rectangles consisting of n lines each containing m characters '.' or '#', which represents the "layers" of the plate in the order from the top to the bottom. The rectangles are separated by empty lines (see the samples). The last line contains x and y $(1 \le x \le n, 1 \le y \le m)$ which are the tap's coordinates. x is the number of the line and y is the number of the column. Lines of each layer are numbered from left to right by the integers from 1 to n, columns of each layer are numbered from top to bottom by the integers from 1 to m.

Output

The answer should contain a single number, showing in how many minutes the plate will be filled.

Examples

input

```
1 1 1

.

1 1
```

output

```
1
```

input

```
2 1 1

.

#

1 1
```

output

```
1
```

input

```
2 2 2

.#
```

```
##

..
..

1 1
```

output

```
5
```

input

```
3 2 2

#.
##

#.
.#

..
..

1 2
```

output

```
7
```

input

```
3 3 3

.#.
###
##.

.##
###
##.

...
...
...

1 1
```

output

```
13
```