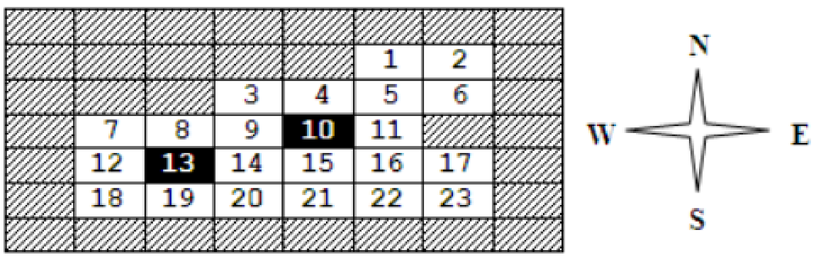
Year 3000. Scientists found a mysterious space station. After months of work, they built a digital map of the station and sent a robot there for further investigation.

The map is an $n \times m$ grid, each cell is either empty or an obstacle. The robot can move east, south, west or north one cell at a time. There is no light or other sensible substance in the space station, so the robot had to "blind walk". Only if he fails to walk, it knows that there is an obstacle in front.

The map is special: all the obstacles are connected (via north, east, west, south directions), and all the empty cells are enclosed by the obstacles. Each two empty cells can be reached from each other, and there is no "tunnels" in the space station (i.e. for each empty cell, at least one of its north neighbor and south neighbor is empty, and at least one of its east neighbor and west neighbor is empty).

We number all empty cells 1, 2, 3, ... from north to south, west to east like this:



There are also k teleport controllers, each controller is connected with a pair of different empty cells (called teleport cell). Each teleport cell can be connected to at most one controller, and the 8 neighbors of each teleport cell will not be another teleport cell or an obstacle.

If two teleport cells are connected to the same controller, once the robot walks into one of them, it'll be teleported to the other cell, then move into the neighbor cell in the same direction. During the process, the robot is not aware of being teleported.

The scientists have already sent a robot to a particular cell. Due to technology restriction, the starting cell is always an empty cell that has a neighboring obstacle. For example, the robot can be sent to cell 12 in figure 1.

If there is no teleport controller, when the robot moves EN, it'll reach cell 8. When he tries to move N, it'll be blocked by an obstacle. However, if a teleport controller connects cell 10 and 13, the robot can successfully execute commands ENN. The actual route is $12 \rightarrow (13 \rightarrow 10 \rightarrow)11 \rightarrow 5 \rightarrow 1$.

Your task is to control the robot to discover all the teleport controllers within a reasonable amount of commands.

Interaction Protocol

Your program should read from standard input, and write to standard output. After printing each line to the standard output, you should flush the output, by calling `fflush(stdout)` *or* `cout << flush` *in C/C++,* `flush(output)` *in Pascal and* `System.out.flush()` *in Java. Please read general instructions for interactive problems for more information.*

There will be at most 20 test cases. For each test case, first read three integers n, m and k ($6 \leq n$, $m \leq 15$, $1 \leq k \leq 5$). Then read n lines, each contains m characters. '`.`' denotes empty cells, '`*`' denotes obstacles, and '`S`' denotes the starting cell. Then issue one or more '`MoveRobot`' command and read the result. When you're ready, issue exactly k '`Answer`' commands.

Command	Description
`MoveRobot` D	Try to move to direction D. return '1' if successful, '0' otherwise.
`Answer` $pos1$ $pos2$	You found a teleport controller connecting empty cells $pos1$ and $pos2$. This command does not return anything.

Note that each teleport controller should be mentioned exactly once, but can be in any order.

The interaction ends when $n = m = k = 0$.

If your program violated any of these rules (bad format, invalid arguments etc), the server will exit immediately, and you will receive Protocol Violation (PV).

Protocol Limit

For each test case, you can issue at most 32767 '`MoveRobot`' commands, **otherwise you'll get Protocol Limit Exceeded (PLE).**

Sample Interaction

```
7 8 1
********
*****..*
***....*
*.....**
*S.....*
*......*
********
                         MoveRobot E
1
                         MoveRobot N
1
                         MoveRobot N
1
                         MoveRobot W
0
                         MoveRobot N
0
                         MoveRobot E
1
                         MoveRobot E
0
                         Answer 10 13
0 0 0
```