Random Mazes

Time limit: *3000 ms*
Memory limit: *256 MB*

We are building a system to control the random construction of mazes for a children's park. There is a robot monster that wanders the maze, but should never be able to reach the child. In addition, there is an added bonus if the child is able to reach a location with a gift toy. The system will ensure that our mazes meet all of the following conditions:

- The robot monster cannot reach the child.
- The maze does not contain any loops in which the child might get stuck.

Our maze will be constructed from an n-by-n grid of rooms, with a starting point for the child `C` in the top left corner, a starting point for the robot monster in the top-right corner, `M`, and a location of the toy `T` in the bottom right. The rows and columns are 0-indexed. For example, here is an example 4x4 grid, with the row and column labels outside the grid:

```
      0   1   2   3
    -----------------
  0 | C |   |   | M |
    -----------------
  1 |   |   |   |   |
    -----------------
  2 |   |   |   |   |
    -----------------
  3 |   |   |   | T |
    -----------------
```

The randomized maze construction system will repeatedly propose randomly selected walls to be removed. We will control whether the removal succeeds by outputting one of the following decisions:

- L: The removal is not allowed because it creates a loop in which the child might get stuck.
- M: The removal is not allowed because the robot monster would be able to reach the child.
- K: The removal is allowed because the robot monster cannot reach the child, and the child does not have any loops in which it could get stuck. However, the child cannot reach the toy.
- T: The removal is allowed because the robot monster cannot reach the child, the child does not have any loops in which it could get stuck, and the child can reach the toy.

In case that multiple conditions apply, use the following rules to break ties:

1. Any proposed removal that allows the robot monster to reach the child will be rejected, and we will output M. This is true even if a loop is created and/or the removal would also allow the child to reach its toy.
2. As long as rule 1 does not apply, any proposed removal that allows the child to reach its toy but also allows the child to get stuck in a loop will be rejected, and we will output L.
3. We will accept moves that do not allow the robot monster to reach the child and do not introduce a loop for the child to get stuck in, even if the move makes it impossible for the child to ultimately reach the toy. For example, all of the scenarios below contain moves that are acceptable:

```
      0   1   2   3
    -----------------
  0 | C |   |   | M |
    -------------   -
  1 |   |   |   |   |
    -------------   -
  2 |   |   |   |   |
    -------------   -
  3 |   |   |   | T |
    -----------------
```

The maze above is acceptable, even though there is no way for the child to safely reach the toy.

```
      0   1   2   3
    -----------------
  0 | C |   |   | M |
    -----------------
  1 |               |
    -----   -----   -
  2 |   |           |
    -------------   -
  3 |   |   |   | T |
    -----------------
```

The maze above is acceptable, even though there is no way for the child to reach the toy, since it would have to pass through a region with a loop.

```
      0   1   2   3
    -----------------
  0 | C |   |   | M |
    -----------------
  1 |   |   |   |   |
    -----------------
  2 |   |   |       |
    ---------   -   -
  3 |   |   |     T |
    -----------------
```

Similarly, the maze above is acceptable, even though there is no way for the child to reach the toy, since the toy is located in a region with a loop.

Standard input

Each input has a single test case.

The input begins with a line containing two space-separated integers, n and w. n specifies the length and width of the maze, and w specifies how many wall removals the system will propose.

Then come w lines, each representing a proposed wall to remove, in the following format: `r c DIR`

This proposal represents a proposal to remove a wall from the room at row r and column c. DIR will be chosen from the set $\{N, E, W, S\}$, indicating which wall should be removed. N, E, W, and S correspond to the north, east, west, or south wall (respectively). North corresponds to the top of the maze, East to the right of the maze, and so on.

Standard output

The output will contain w lines, containing the w responses to the maze creation system's proposals. Each line will contain the appropriate symbol chosen from the set: {`L`, `M`, `K`, `T`}.

Constraints and notes

- $2 \leq N \leq 500$
- $1 \leq w \leq 400,000$
- $0 \leq r, c \leq N-1$

The maze construction system will never propose that an outer wall should be removed. For example, it would not propose `0 0 N`

The system will never propose that the same wall should be removed twice. It would not, for example, ask to remove the east wall of room (0,0) twice. Similarly, it would not ask to remove the east wall of room (1,1), and then ask to remove the west wall of room (1,2), since this is the same wall.

Input

```
4 19
1 3 S
2 2 N
2 2 E
1 2 E
0 0 E
0 1 E
0 2 E
3 0 E
0 2 S
3 3 W
0 3 S
3 2 W
1 0 N
1 0 S
3 0 N
1 1 W
1 1 E
1 1 S
3 1 N
```

Output

```
K
K
K
K
K
K
M
K
L
K
K
K
K
K
T
T
M
T
L
```

Proposal 1: `1 3 S`

After breaking down the South wall at (1,3), the maze looks like:

```
      0   1   2   3
    -----------------
  0 | C |   |   | M |
    -----------------
  1 |   |   |   |   |
    -------------   -
  2 |   |   |   |   |
    -----------------
  3 |   |   |   | T |
    -----------------
```

No constraints are violated, so we output: K

Proposal 2: `2 2 N`

After breaking down the North wall at (2,2), the maze looks like:

```
      0   1   2   3
    -----------------
  0 | C |   |   | M |
    -----------------
  1 |   |   |   |   |
    ---------   -   -
  2 |   |   |   |   |
    -----------------
  3 |   |   |   | T |
    -----------------
```

No constraints are violated, so we output: K

Proposal 3: `2 2 E`

After breaking down the East wall at (2,2), the maze looks like:

```
      0   1   2   3
    -----------------
  0 | C |   |   | M |
    -----------------
  1 |   |   |   |   |
    ---------   -   -
  2 |   |   |       |
    -----------------
  3 |   |   |   | T |
    -----------------
```

No constraints are violated, so we output: K

Proposal 4: `1 2 E`

After breaking down the East wall at (1,2), the maze looks like:

```
      0   1   2   3
    -----------------
  0 | C |   |   | M |
    -----------------
  1 |   |   |       |
    ---------   -   -
  2 |   |   |       |
    -----------------
  3 |   |   |   | T |
    -----------------
```

Note that there is a loop in the maze, but the child cannot reach the loop, so it is ok. No constraints are violated, so we output: K

Proposal 5: `0 0 E`

After breaking down the East wall at (0,0), the maze looks like:

```
      0   1   2   3
    -----------------
  0 | C     |   | M |
    -----------------
  1 |   |   |       |
    ---------   -   -
  2 |   |   |       |
    -----------------
  3 |   |   |   | T |
    -----------------
```

No constraints are violated, so we output: K

Proposal 6: `0 1 E`

After breaking down the East wall at (0,1), the maze looks like:

```
      0   1   2   3
    -----------------
  0 | C         | M |
    -----------------
  1 |   |   |       |
    ---------   -   -
  2 |   |   |       |
    -----------------
  3 |   |   |   | T |
    -----------------
```

No constraints are violated, so we output: K

Proposal 7: `0 2 E`

If we were to break down the East wall at (0,2), the maze would look like:

```
      0   1   2   3
    -----------------
  0 | C           M |
    -----------------
  1 |   |   |       |
    ---------   -   -
  2 |   |   |       |
    -----------------
  3 |   |   |   | T |
    -----------------
```

Note that the robot monster would be able to get to the child, so we reject this proposal. The maze looks exactly like it did after **Proposal 6** and we output: M

Proposal 8: `3 0 E`

After breaking down the East wall at (3,0), the maze looks like:

```
      0   1   2   3
    -----------------
  0 | C         | M |
    -----------------
  1 |   |   |       |
    ---------   -   -
  2 |   |   |       |
    -----------------
  3 |       |   | T |
    -----------------
```

No constraints are violated, so we output: K

Proposal 9: `0 2 S`

If we were to break down the South wall at (0,2), the maze would look like:

```
      0   1   2   3
    -----------------
  0 | C         | M |
    ---------   -----
  1 |   |   |       |
    ---------   -   -
  2 |   |   |       |
    -----------------
  3 |       |   | T |
    -----------------
```

Note that the child could get caught in a loop here, so we do not allow this change. The maze looks exactly like it did after **Proposal 8** and we output: L

Proposal 10: `3 3 W`

After breaking down the West wall at (3,3), the maze looks like:

```
      0   1   2   3
    -----------------
  0 | C         | M |
    -----------------
  1 |   |   |       |
    ---------   -   -
  2 |   |   |       |
    -----------------
  3 |       |     T |
    -----------------
```

No constraints are violated, so we output: K

Proposal 11: `0 3 S`

After breaking down the South wall at (0,3), the maze looks like:

```
      0   1   2   3
    -----------------
  0 | C         | M |
    -------------   -
  1 |   |   |       |
    ---------   -   -
  2 |   |   |       |
    -----------------
  3 |       |     T |
    -----------------
```

Note that it is ok to have a loop in the maze that the robot monster might get stuck in. No constraints are violated, so we output: K

Proposal 12: `3 2 W`

After breaking down the West wall at (3,2), the maze looks like:

```
      0   1   2   3
    -----------------
  0 | C         | M |
    -------------   -
  1 |   |   |       |
    ---------   -   -
  2 |   |   |       |
    -----------------
  3 |             T |
    -----------------
```

No constraints are violated, so we output: K

Proposal 13: `1 0 N`

After breaking down the North wall at (1,0), the maze looks like:

```
      0   1   2   3
    -----------------
  0 | C         | M |
    -   ---------   -
  1 |   |   |       |
    ---------   -   -
  2 |   |   |       |
    -----------------
  3 |             T |
    -----------------
```

No constraints are violated, so we output: K

Proposal 14: `1 0 S`

After breaking down the South wall at (1,0), the maze looks like:

```
      0   1   2   3
    -----------------
  0 | C         | M |
    -   ---------   -
  1 |   |   |       |
    -   -----   -   -
  2 |   |   |       |
    -----------------
  3 |             T |
    -----------------
```

No constraints are violated, so we output: K

Proposal 15: `3 0 N`

After breaking down the North wall at (3,0), the maze looks like:

```
      0   1   2   3
    -----------------
  0 | C         | M |
    -   ---------   -
  1 |   |   |       |
    -   -----   -   -
  2 |   |   |       |
    -   -------------
  3 |             T |
    -----------------
```

No constraints are violated _and_ the child can reach the toy, so we output: T

Proposal 16: `1 1 W`

After breaking down the West wall at (1,1), the maze looks like:

```
      0   1   2   3
    -----------------
  0 | C         | M |
    -   ---------   -
  1 |       |       |
    -   -----   -   -
  2 |   |   |       |
    -   -------------
  3 |             T |
    -----------------
```

No constraints are violated _and_ the child can reach the toy, so we output: T

Proposal 17: `1 1 E`

If we were to break down the east wall at (1,1), the maze would look like:

```
      0   1   2   3
    -----------------
  0 | C         | M |
    -   ---------   -
  1 |               |
    -   -----   -   -
  2 |   |   |       |
    -   -------------
  3 |             T |
    -----------------
```

The child could get to the toy, but they could get stuck in a loop and the robot monster could reach the child. In this case, we reject the proposal. The maze will be exactly as it was after **Proposal 16**. We output (see notes in the Introduction about breaking ties like this): M

Proposal 18: `1 1 S`

After breaking down the South wall at (1,1), the maze looks like:

```
      0   1   2   3
    -----------------
  0 | C         | M |
    -   ---------   -
  1 |       |       |
    -   -   -   -   -
  2 |   |   |       |
    -   -------------
  3 |             T |
    -----------------
```

No constraints are violated _and_ the child can reach the toy, so we output: T

Proposal 19: `3 1 N`

If we were to break down the North wall at (3,1). The maze would look like:

```
      0   1   2   3
    -----------------
  0 | C         | M |
    -   ---------   -
  1 |       |       |
    -   -   -   -   -
  2 |   |   |       |
    -   -   ---------
  3 |             T |
    -----------------
```

While the child could reach the toy, they could also get stuck in a loop. Thus, we reject this proposal, and output: L