# 02. Mouse In The Kitchen



*A hungry little mouse is living in an old suburbs house. It walks around the kitchen cupboard every night and eats all the cheese. A lazy plump cat is guarding the kitchen, so the mouse should not walk out of the cupboard area.*

In the beginning, you will be given **N** and **M** – **integers, separated by a comma - ","**, indicating the **cupboard’s area dimensions.** On the next **N** lines, you will receive strings, representing the **rows** of the area, with **M** columns.

After that, on each line, **until** the command **"danger"** appears as an input string, you will receive the **possible directions** for the mouse to move - **"up", "down", "right",** and **"left"**.

If the mouse **steps outside the cupboard area**, the cat will attack, and the **cheese hunt is over**. In that case, **the program ends**, **keep** the last known position of the mouse, before it steps outside the cupboard, and the following **message** is printed on the Console: **"No more cheese for tonight!"**

**Possible characters in the matrix** are:

* **M** - represents the mouse's position.
* **C** – represents a piece of cheese.
* **\*** – represents an empty position, nothing happens if the mouse steps on it.
* **@** – represents a wall in the cupboard, cannot step on or go through it.
* **T** – represents a trap.

The mouse starts from the **M - position**.

* If the mouse steps on **C – position**, it **eats the cheese** from the field, and the **position** **is marked** with **"**\***".** 
  + If this is **the last cheese** in the cupboard area, the mouse goes to sleep. Remember that this will be **the last known position of the mouse**. **The program ends** and the following **message** is printed on the Console: **"Happy mouse! All the cheese is eaten, good night!"**
* If the mouse steps into a **trap (T -position)**, it will be **trapped**. Remember that this will be **the last known position of the mouse**. In that case, the **program ends**, and the following **message** is printed on the Console: **"Mouse is trapped!"**
* If the **given direction** **leads** the mouse towards **@ - position,** this is a wall in the cupboard area. **Do not make the move** and **skip the command**.
* If the **"danger"** command is received **before** the mouse manages to **eat all the cheese**, the mouse disappears. Remember that this will be **the last known position of the mouse and you will need it for the final state of the matrix**. In that case, the **program ends**, and the following **message** is printed on the Console: **"Mouse will come back later!"**

In the end, **print the final state of the matrix** (cupboard area) with **the last known position of the mouse** in it. **Each row on a new line**.

## Input

* **On the first line** you will get the **number of rows** and **columns of the matrix, separated by a comma**.
* On the next **N** lines, you will receive strings, representing each **row** of the matrix.
* On each of the following lines, until the command **"danger"** appears as an input string, you will receive the possible directions for the mouse to move - **"up", "down", "right",** and **"left"**.
* **"danger"** command – The mouse spots danger and disappears… for now!

## Output

* **On the first line:**
  + If the mouse **steps outside the cupboard** – **"No more cheese for tonight!"**.
  + If the mouse manages to eat all the cheese –**"Happy mouse! All the cheese is eaten, good night!"**.
  + If the mouse steps into a **trap (T -position)** –"**Mouse is trapped!**".
  + If the **"danger"** command is received before the mouse manages to eat all the cheese –

**"Mouse will come back later!"**.

* On the next lines, print the **final state of the matrix** with the **last known position of the mouse** in it. **Each row - on a new line**.

## Constraints

* There will always be **at least one trap** in the cupboard.
* There **will always be some cheese** in the cupboard.
* There will always be a **"danger"** command in the end, but **it is not necessary to reach it**. **The program may end earlier**.
* **Each row** of the matrix will have the **same length**.

## Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 5,5  \*\*M\*\*  T@@\*\*  CC@\*\*  \*\*@@\*  \*\*CC\*  left  down  left  down  down  down  right  danger | Mouse is trapped!  \*\*\*\*\*  M@@\*\*  CC@\*\*  \*\*@@\*  \*\*CC\* | The mouse moves to the left and the position is marked with **"\*"**, so nothing happens.  The next command is **"down"**, but the position is marked with **"@"**, so we skip the command.  The next command - **"left"**, the mouse moves to the left. The position is marked with **"\*"**, so nothing happens.  The next command is **"down"**, the position is marked with **"T"**, so the mouse is trapped. Remember to mark the last known position of the mouse with **"M"**. |
| 4,8  CC@\*\*C\*M  T\*@\*\*CT\*  \*\*@@@@\*\*  T\*\*\*C\*\*\*  down  right  left  down  left  danger | No more cheese for tonight!  CC@\*\*C\*\*  T\*@\*\*CTM  \*\*@@@@\*\*  T\*\*\*C\*\*\* |  |
| 6,3  @CC  @TC  @C\*  @M\*  @\*\*  @\*\*  left  up  left  right  up  up  left  left  danger | Happy mouse! All the cheese is eaten, good night!  @M\*  @T\*  @\*\*  @\*\*  @\*\*  @\*\* |  |