Code Jam 2012 - Round 2

Descending in the Dark

Problem

You are on the face of Mount Everest. You need to find shelter before you freeze, and it's dark! What do you do?

The good news is you have already memorized the layout of the mountain. It is a grid with certain squares impassable and other squares containing caves where you can rest for the night. The bad news is you don't know where you are, and it's too steep to climb up. All you can do is move left, right, or down.

Here is an example layout, with '.' representing a passable square, '#' representing an impassable square, and numbers representing caves.

```
######
##...#
#..#.#
#...##
#0#..#
####1#
######
```

Since it is so dark, you will move around by following a *plan*, which is a series of instructions, each telling you to move one square left, right, or down. If an instruction would take you to a passable square or to a cave, you will follow it. If it would take you to an impassable square, you will have to ignore it. Either way, you will continue on to the next step, and so on, until you have gone through the whole plan.

To help with your descent, you want to find out two things for each cave **C**:

- What squares is it possible to reach **C** from? We will label the set of these squares by $\mathbf{S_C}$, and the number of them by $\mathbf{n_C}$.
- Is there a single plan that, if followed from any square in $\mathbf{S_C}$, will finish with you at cave **C**? If so, we say the cave is *lucky*.

Note that you might pass by several caves while following a plan. All that matters is what square you *finish* on after executing all the steps, not what caves you visit along the way.

For example, in the layout above, cave 0 is lucky. There are 9 squares that it can be reached from (including itself), and the plan "left-left-down-down-left-down" will finish with you at the cave from any of those squares.

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow, beginning with a line containing integers **R** and **C**, representing the number of rows and columns in the mountain layout.

This is followed by **R** lines, each containing **C** characters, describing a mountain layout. As in the example above, a '#' character represents an impassable square, a '.' character represents a passable square, and the digits '0'-'9' represent caves (which are also passable squares).

Output

For each test case, first output one line containing "Case #x:", where x is the case number (starting from 1). For each cave **C**, starting with 0 and counting up from there, write a line "**C**: $\mathbf{n_C}$ $\mathbf{L_C}$". Here, **C** is the cave number, $\mathbf{n_C}$ is the number of squares you can reach the cave from, and $\mathbf{L_C}$ is either the string "Lucky" or the string "Unlucky", as defined above.

Limits

Memory limit: 1GB.
Time limit: 40 seconds per test set.
There will be between 1 and 10 caves inclusive.
If there are *d* caves, they will be labeled with the digits {0, 1, ..., *d* - 1}, and no two caves will have the same label.
All squares on the boundary of the mountain layout will be impassable.
$1 \leq \mathbf{T} \leq 20$.

Test set 1 (Visible Verdict)

$3 \leq \mathbf{R}, \mathbf{C} \leq 10$.

Test set 2 (Hidden Verdict)

$3 \leq \mathbf{R}, \mathbf{C} \leq 60$.

Sample

Sample Input

```
2
7 5
#####
##0##
##1.#
##2##
#3..#
#.#.#
#####
7 6
######
##...#
#..#.#
#...##
#0#..#
####1#
######
```

Sample Output

```
Case #1:
0: 1 Lucky
1: 3 Lucky
2: 4 Unlucky
3: 7 Lucky
Case #2:
0: 9 Lucky
1: 11 Unlucky
```

In the first case, here are some valid plans you could use for the lucky caves:

- For cave 0, you can use the empty plan. If you can reach the cave at all, you are already in the right place!
- For cave 1, you can use the plan right-down-left.
- For cave 3, you can use the plan right-right-left-down-down-down-left.