Code Jam 2017 - Round 2

Beaming With Joy

Problem

Joy is about to go on a long vacation, so she has hired technicians to install a security system based on infrared laser beams. The technicians have given her a diagram that represents her house as a grid of unit cells with **R** rows and **C** columns. Each cell in this grid contains one of the following:

- `/`: A two-sided mirror that runs from the cell's lower left corner to its upper right corner.
- `\`: A two-sided mirror that runs from the cell's upper left corner to its lower right corner.
- `-`: A beam shooter that shoots horizontal beams out into the cells (if any) to the immediate left and right of this cell.
- `|`: A beam shooter that shoots vertical beams out into the cells (if any) immediately above and below this cell.
- `#`: A wall. (Note that the house is not necessarily surrounded by a border of walls; this is one reason why Joy needs a security system!)
- `.`: Nothing; the cell is empty.

Beams travel in straight lines and continue on through empty cells. When a beam hits a mirror, it bounces 90 degrees off the mirror's surface and continues. When a beam traveling to the right hits a `/` mirror, it bounces off the mirror and starts traveling up; beams traveling up, left, or down that hit a `/` mirror bounce off and travel right, down, or left, respectively. The `\` mirror behaves similarly: when a beam traveling right, up, left or down hits it, it bounces off and starts traveling down, left, up or right, respectively. When a beam hits a wall or goes out of the bounds of the grid, it stops. It is fine for beams to cross other beams, but if a beam hits any beam shooter (including, perhaps, the beam shooter that originated the beam), that beam shooter will be destroyed!

Joy wants to make sure that every empty cell in the house has at least one beam passing through it, and that no beam shooters are destroyed, since that would just be wasting money! Unfortunately, the technicians have already installed the system, so the most Joy can do is rotate some of the existing beam shooters 90 degrees. That is, for any number (including zero) of beam shooters, she can turn `-` into `|` or vice versa.

Can you find any way for Joy to achieve her goal, or determine that it is impossible? Note that it is *not* required to minimize the number of rotations of beam shooters.

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each case begins with one line with two integers **R** and **C**: the number of rows and columns in the grid representing the house. Then, **R** lines of **C** characters each follow; each character is `/`, `\`, `-`, `|`, `#`, or `.`, as described in the statement.

Output

For each test case, output one line containing `Case #x: y`, where `x` is the test case number (starting from 1) and `y` is `IMPOSSIBLE` if Joy cannot accomplish her goal, or `POSSIBLE` if she can. Then, if the case is possible, output the same **R** lines of **C** characters each from the input grid, with zero or more instances of `-` replaced by `|` or vice versa.

If there are multiple possible answers, you may output any of them.

Limits

Time limit: 20 seconds per test set.
Memory limit: 1 GB.
1 ≤ **T** ≤ 100.
1 ≤ **C** ≤ 50.
Each character in the grid is one of `/`, `\`, `-`, `|`, `#`, or `.`.
The number of `-` characters plus the number of `|` characters (that is, the number of beam shooters) in the grid is between 1 and 100, inclusive.
There is at least 1 `.` character (that is, empty space) in the grid.

Small dataset (Test Set 1 - Visible)

1 ≤ **R** ≤ 5.
There are no `/` or `\` characters (that is, no mirrors) in the grid.

Large dataset (Test Set 2 - Hidden)

1 ≤ **R** ≤ 50.

Sample

Sample Input

```
5
1 3
-.-
3 4
#.##
#--#
####
2 2
-.
#|
4 3
.|.
-//
.-.
#\/
3 3
/|\
\\/
./#
```

Sample Output

```
Case #1: IMPOSSIBLE
Case #2: POSSIBLE
#.##
#||#
####
Case #3: POSSIBLE
|.
#|
Case #4: POSSIBLE
.-.
|//
.|.
#\/
Case #5: IMPOSSIBLE
```

Note that the last 2 sample cases would not appear in the Small dataset.

In Sample Case #1, if a beam shooter is positioned to shoot its beam into the empty cell, it will necessarily destroy the other beam shooter. So the case is `IMPOSSIBLE`.

In Sample Case #2, the leftmost beam shooter must be rotated to cover the empty cell. The rightmost beam shooter must also be rotated to avoid destroying the leftmost beam shooter.

In Sample Case #3, the existing beam shooters already cover all empty cells with their beams and do not destroy each other, so outputting the grid from the input would be acceptable. However, notice that the output that we have given is also correct.

In Sample Case #4, one acceptable solution is to rotate all three of the beam shooters. However, note that the following would also be acceptable:

```
.-.
|//
.-.
#\/
```

since it is not necessary for cells with mirrors to have a beam pass through them. (Who would steal giant diagonal mirrors, anyway?)

In Sample Case #5, the beam shooter would destroy itself no matter which orientation Joy chooses for it, so the case is `IMPOSSIBLE`.