Coding Competitions Farewell Rounds - Round D

Hey Google, Drive!

Problem

The Google Assistant and Android Auto teams are collaborating on a new prototype car that can be driven via voice commands. The early prototype works through a phone connected to a car simulator. Unfortunately, one of the early testers dropped their phone in the toilet, damaging the microphone and making it harder to use the new feature. Since they do not want to miss out on the opportunity, they want your help to use it anyway.

The early prototype moves on a simple grid of $\mathbf{R}$ rows and $\mathbf{C}$ columns and only understands 4 very simple voice commands: north, south, east, and west. Each command makes the car try to move exactly one cell in the corresponding direction. Because of the microphone issues, however, the system may mishear and interchange north and south, and separately, east and west. That means that a command of north may make the car move north or south, a command of south may make the car move south or north, and similarly both commands east and west may make the car move east or west when issued. In all cases, both movement options can happen with equal probability ($1/2$).

The tester set up a driving grid such that each cell can contain either a wall, a hazard, or be empty. If a command would make the car move into a wall, or outside the grid, it does nothing instead. If a command makes the car move into a hazard, the car cannot execute any more commands.

The tester has marked some empty cells of the grid as interesting starts and others as interesting finishes. A pair of an interesting start and an interesting finish is *drivable* if there is a strategy to drive the car through voice commands from the start that makes it end at the finish with probability at least $1 - 10^{-10^{100}}$. A strategy can choose which command to issue and when to stop depending on the outcome of the previous commands. Notice that if the car moves into a hazard it stops moving, so it cannot make it to the finish. The tester wants your help finding the list of all drivable pairs.

Input

The first line of the input gives the number of test cases, $\mathbf{T}$. $\mathbf{T}$ test cases follow. Each test case starts with a line containing two integers $\mathbf{R}$ and $\mathbf{C}$, the number of rows and columns of the grid. Then, $\mathbf{R}$ lines follow containing a string of $\mathbf{C}$ characters each. The j-th character on the i-th of these lines $\mathbf{G_{i,j}}$ represents the grid in the i-th row and j-th column as follows:

- A period (`.`) represents an uninteresting empty cell.
- A hash symbol (`#`) represents a cell containing a wall.
- An asterisk (`*`) represents a cell containing a hazard.
- An English lowercase letter (`a` through `z`) represents an empty cell that is an interesting start.
- An English uppercase letter (`A` through `Z`) represents an empty cell that is an interesting finish.

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 `NONE` if there are no drivable pairs. Otherwise, y must be a series of 2

character strings separated by spaces, representing all drivable pairs with the start letter first and the finish letter second, in alphabetical order.

Limits

Time limit: 60 seconds.
Memory limit: 2 GB.
$1 \leq \mathbf{T} \leq 100$.
$\mathbf{G_{i,j}}$ is either a period (`.`), a hash symbol (`#`), an asterisk (`*`) or a lowercase or uppercase English letter, for all i, j.
The set $\{\mathbf{G_{i,j}}$ for all $i, j\}$ contains at least 1 lowercase and at least 1 uppercase English letter.
Each lowercase and uppercase letter appears at most once among all $\mathbf{G_{i,j}}$.

Test Set 1 (Visible Verdict)

$1 \leq \mathbf{R} \leq 20$.
$1 \leq \mathbf{C} \leq 20$.

Test Set 2 (Hidden Verdict)

$1 \leq \mathbf{R} \leq 100$.
$1 \leq \mathbf{C} \leq 100$.

Sample

Sample Input

```
4
1 2
aZ
4 4
a..c
**.*
.Y.#
bX#Z
2 2
a*
*Z
2 7
a*bcd*.
...*F#.
```

Sample Output

```
Case #1: aZ
Case #2: aY bX bY cY
Case #3: NONE
Case #4: dF
```

In Sample Case #1, simply repeating the west command until reaching the finish is a viable strategy. Each time there is a $1/2$ probability of reaching the finish and a $1/2$ probability of staying in the same place. Thus, the probability of not reaching the finish in 10^{101} or fewer steps is $2^{-10^{101}} < 10^{-10^{100}}$.

In Sample Case #2 a similar strategy as in Sample Case #1 can be used to get the car from any position in the top row (1) to any other with probability as high as desired, and similarly for all non-wall positions in the third row from the top (2). Analogously, but using the south command, the car can move between non-wall positions on the third column from the left (3).
From both `a` and `c` we can use (1) to get to the third column from the left, then (3) to get right next to `Y` and then (2) to get to `Y` making both `aY` and `cY` drivable. Notice, however, that safely

using the north or south commands from the third row can only be done in the third column, or otherwise the car may go into a hazard. Therefore, there is no safe way to move the car from the third to the fourth row, making `aX` and `cX` not drivable.
From `b`, however, the car can use a similar strategy to get to `X`, and from `X` the car can get to `Y` by using the north or south command repeatedly (and stop when reaching `Y`, never risking going into the hazard above).
Finally, the finish `Z` is completely isolated, so it cannot be part of a drivable pair.

In Sample Case #3, every path from the interesting start to the interesting finish goes through a hazard, which makes the pair not drivable.

In Sample Case #4, only the interesting start `d` has a viable strategy to get to the finish `F`.