Code Jam 2016 - World Finals

Map Reduce

Problem

Ben the brilliant video game designer is trying to design maps for his upcoming augmented-reality mobile game. Recently, he has created a map which is represented as a matrix of **R** rows and **C** columns. The map consists of a bunch of `.` characters representing empty squares, a bunch of `#` characters representing impassable walls, a single start position represented by `S` and a single finish position represented by `F`. For example, the map could look like:

```
############
#S..#..##...#
###.##..#.#F#
#...##.##.###
#.#.........#
############
```

In Ben's game, a *path* is a sequence of steps (up, down, left or right) to go from one cell to another while not going through any impassable walls.

Ben considers a *good* map to have the following properties:

- There is a path between any two empty squares (including the start and finish positions).
- To preserve structural integrity, impassable walls must meet at edges and not just corners. For every 2×2 region in the map, if the region contains exactly two walls, then those walls are either in the same row or the same column. In other words, there is no 2×2 region where the walls are in one of these two configurations:

  ```
  #.    .#
  .#    #.
  ```

- The boundary consists of only impassable walls. A cell is considered part of the boundary if it is in the uppermost/lowermost rows or if it is in the leftmost/rightmost columns.

The distance of the shortest path is the minimum number of steps required to reach the finish position from the start position. For instance, the shortest path in the above example takes 17 steps.

Being such a clever mapmaker, Ben realized that he has created a map that is too hard for his friends to solve. He would like to reduce its difficulty by removing some of the impassable walls. In particular, he wants to know whether it is possible to remove zero or more impassable walls from his map such that the shortest path from start to finish takes *exactly* **D** steps, and that the resulting map is still *good*. Note that it is not enough to simply find *a* path with **D** steps; **D** must be the number of steps in the *shortest* path.

For example, if **D** = 15, we could remove the impassable wall directly below the finish position to get a good solution.

```
############
#S..#..##...#
###.##..#.#F#
#...##.##.#.#
```

```
#.#.........#
############
```

There is no solution if **D** = 5.

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each test case starts with a line containing three space-separated integers **R**, **C** and **D**: the number of rows and columns in the map, and the desired number of steps in the shortest path from start to finish after possibly removing impassable walls. **R** lines follow, each consisting of **C** characters (either `.`, `#`, `S` or `F`) representing Ben's map.

It is guaranteed that the map is good, as described in the problem 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 the word `POSSIBLE` or `IMPOSSIBLE`, depending on whether the shortest path can be made equal to **D** by removing some number of walls such that the map is still good. If it is possible, output **R** more lines containing **C** characters each, representing the new map. In your output, replace the `#` characters for removed walls (if any) with `.` characters.

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

Limits

Memory limit: 1 GB.
$1 \leq \mathbf{T} \leq 100$.
Each test case contains exactly one `S` and exactly one `F`.
The input file is at most 3MB in size.

Small dataset (Test Set 1 - Visible)

Time limit: 60 seconds.
$3 \leq \mathbf{R} \leq 40$.
$3 \leq \mathbf{C} \leq 40$.
$1 \leq \mathbf{D} \leq 1600$.

Large dataset (Test Set 2 - Hidden)

Time limit: 300 seconds.
$3 \leq \mathbf{R} \leq 1000$.
$3 \leq \mathbf{C} \leq 1000$.
$1 \leq \mathbf{D} \leq 10^6$.

NOTE: The Large output breaks the usual cap on Code Jam output size, but you can upload it as normal.

Sample

Sample Input	Sample Output

```
3
6 13 15
#############
#S..#..##...#
###.##..#.#F#
#...##.##.###
#.#.........#
#############
5 8 3
########
#S.....#
####...#
#F.....#
########
4 10 11
##########
#S#...#.F#
#...#...##
##########
```

```
Case #1: POSSIBLE
#############
#S..#..##...#
###.##..#.#F#
#...##.##.#.#
#.#.........#
#############
Case #2: IMPOSSIBLE
Case #3: POSSIBLE
##########
#S#...#.F#
#...#...##
##########
```

The sample output displays one set of answers to the sample cases. Other answers may be possible.

Sample case #1 is the example in the problem statement.

In sample case #2, it is possible to remove walls to make the distance of the shortest path either 2 or 4, for example. However, there is no way to make the distance of the shortest path exactly 3.

In sample case #3, the shortest path already takes 11 steps to begin with, so there is no need to reduce the difficulty of the map.