

Bomberdude

Filename: bomb

The Problem

You've been playing the new hit videogame Bomberdude, but aren't doing very well and keep dying before you make it to the end. The game is played on a 2D maze. You always start in the top left and your goal is to make it to the bottom right. Both of these spots will always be open. But of course, it's not that easy. At the beginning of the map, you can't see any of the bombs, because they're all buried, waiting to be stepped on and exploded and decrease your health. There are also walls sporadically placed on the map to force your movement at some points.

But hope is not lost. Through various methods which shan't be named here, you've obtained the locations of the bombs for some maps, in order to help you plan out the best route to make it to the exit. Your maps also give the 'power' of each bomb, an integer which denotes how far in each cardinal direction the bomb explodes. For example, a bomb with power 2 will affect 9 spots on the grid: the spot on which the bomb is buried, the 2 spots to the left of the bomb, the 2 spots to the north of the bomb, the 2 spots to the east of the bomb, and the 2 spots to the south of the bomb. Bombs with power 0 will affect only the spot on which the bomb is located.

Bombs will affect everything around them. Each bomb whose blast touches Bomberdude will decrease his health by 1. If Bomberdude's health reaches 0, he is dead. Bombs that hit walls will cause them to crumble and open that spot for walking. Bombs whose blast hits other bombs will cause those bombs to detonate, the same as if they were stepped on. Notice that this could cause chain reactions of detonations.

Your goal is to write a program to determine if your planned path will get you to the exit, given the location and power of all bombs, the location of all walls, and your starting health.

The Input

Then input will begin with a positive integer n indicating the number of mazes to solve. Each maze will begin with a line contain three positive integers, $1 \leq r \leq 20$, the number of rows in the maze, $1 \leq c \leq 20$, the number of columns in the maze, and $1 \leq h \leq 20$, the starting health of Bomberdude. Following this will be r lines, each of length c , which gives the maze with marked locations. A '.' indicates an open spot, a 'W' indicates a wall, and an integer between 0 and 9 (inclusive) indicates a bomb with that power. The last line gives the sequence of moves you have planned out, which will consist of the characters L R U and D for left, right, up, and down. Remember that Bomberdude can't walk through walls, so any move that attempts to push through one should be ignored. You can also imagine a wall around the entire boundary of the maze, so you can never leave the defined grid. Keep in mind that once you reach the exit, any other moves you may have planned out will be ignored.

The Output

For each maze, you should first output “Maze *i*: ” (beginning at 1), followed by one of these three lines, depending on how far Bomberdude makes it:

```
You made it to the end. Great job!
Oh no, you exploded!
You didn't die, but you didn't win either. Lamé!
```

Clearly, you should only output the first if you safely make it to the exit, the second if your health reaches 0 before you make it to the exit, and the third if your series of moves doesn't kill you, but also doesn't leave you at the exit. Each line of output should be followed by a newline.

Sample Input

```
3
4 4 5
..2W
WW.W
W4.W
....
RRDDDR
4 4 5
..WW
W.WW
W3WW
W2..
DRRDDRR
4 4 5
.113
22..
..0.
WWW.
RRDDRD
```

Sample Output File

```
Maze 1: You made it to the end. Great job!
Maze 2: You didn't die, but you didn't win either. Lamé!
Maze 3: Oh no, you exploded!
```