The Bomberman Game ☆

Bomberman lives in a rectangular grid. Each cell in the grid either contains a bomb or nothing at all.

Each bomb can be planted in any cell of the grid but once planted, it will detonate after *exactly 3 seconds*. Once a bomb detonates, it's destroyed — along with anything in its four neighboring cells. This means that if a bomb detonates in cell i, j, any valid cells $(i \pm 1, j)$ and $(i, j \pm 1)$ are cleared. If there is a bomb in a neighboring cell, the neighboring bomb is destroyed *without* detonating, so there's no chain reaction.

Bomberman is immune to bombs, so he can move freely throughout the grid. Here's what he does:

1. Initially, Bomberman arbitrarily plants bombs in some of the cells, the initial state.
2. After one second, Bomberman does nothing.
3. After one more second, Bomberman plants bombs in all cells without bombs, thus filling the whole grid with bombs. No bombs detonate at this point.
4. After one more second, any bombs planted exactly three seconds ago will detonate. Here, Bomberman stands back and observes.
5. Bomberman then repeats steps 3 and 4 indefinitely.

Note that during every second Bomberman plants bombs, the bombs are planted simultaneously (i.e., *at the exact same moment*), and any bombs planted at the same time will detonate at the same time.

Given the initial configuration of the grid with the locations of Bomberman's first batch of planted bombs, determine the state of the grid after N seconds.

For example, if the initial grid looks like:

```
...
.O.
...
```

it looks the same after the first second. After the second second, Bomberman has placed all his charges:

```
OOO
OOO
OOO
```

At the third second, the bomb in the middle blows up, emptying all surrounding cells:

```
...
...
...
```

Input Format

Author	kevinsogo
Difficulty	Medium
Max Score	40
Submitted By	8041

NEED HELP?

View discussions

View editorial

View top submissions

RATE THIS CHALLENGE

☆ ☆ ☆ ☆ ☆

MORE DETAILS

Download problem statement

Download sample test cases

Suggest Edits

The first line contains three space-separated integers r, c, and n, The number of rows, columns and seconds to simulate.
Each of the next r lines contains a row of the matrix as a single string of c characters. The `.` character denotes an empty cell, and the `O` character (ascii 79) denotes a bomb.

Constraints

- $1 \leq r, c \leq 200$
- $1 \leq n \leq 10^9$

Subtask

- $1 \leq n \leq 200$ for 40% of the maximum score.

Output Format

Print the grid's final state. This means R lines where each line contains C characters, and each character is either a `.` or an `O` (ascii 79). This grid must represent the state of the grid after n seconds.

Sample Input

```
6 7 3
.......
...O...
....O..
.......
OO.....
OO.....
```

Sample Output

```
000.000
00...00
000...0
..00.00
...0000
...0000
```

Explanation

The initial state of the grid is:

```
.......
...O...
....O..
.......
OO.....
OO.....
```

Bomberman spends the first second doing nothing, so this is the state after 1 second:

```
.......
...O...
....O..
.......
OO.....
OO.....
```

Bomberman plants bombs in all the empty cells during his second second, so this is the state after 2 seconds:

```
0000000
0000000
```

```
0000000
0000000
0000000
0000000
```

In his third second, Bomberman sits back and watches all the bombs he planted 3 seconds ago detonate. This is the final state after 3 seconds:

```
000.000
00...00
000...0
..00.00
...0000
...0000
```

Current Buffer (saved locally, editable) Java 7

```
1  import java.io.*;
2  import java.util.*;
3  import java.text.*;
4  import java.math.*;
5  import java.util.regex.*;
6
7  public class Solution {
8
9      static String[] bomberMan(int n, String[] grid) {
10         // Complete this function
11     }
12
13     public static void main(String[] args) {
14         Scanner in = new Scanner(System.in);
15         int r = in.nextInt();
16         int c = in.nextInt();
17         int n = in.nextInt();
18         String[] grid = new String[r];
19         for(int grid_i = 0; grid_i < r; grid_i++){
20             grid[grid_i] = in.next();
21         }
22         String[] result = bomberMan(n, grid);
23         for (int i = 0; i < result.length; i++) {
24             System.out.print(result[i] + (i != result.length - 1 ? "\n"
: ""));
25         }
26         System.out.println("");
27
28
29         in.close();
30     }
31 }
32
```

Line: 1 Col: 1

Upload Code as File Test against custom input Run Code Submit Code