CCO '99 P3 - Manelzuma's Revenge

Time limit: 2.0s **Memory limit:** 32M

Canadian Computing Competition: 1999 Stage 2, Day 1, Problem 3

You may recall that in stage 1 of the CCC you were asked to calculate a region of a fractal obtained by starting with a square, dividing it into 9 sub-squares, then clearing the middle, like so:

```
* * * * * * * * *
* * * * * * * * *
* * * * * * * * *
* * *       * * *
* * *       * * *
* * *       * * *
* * * * * * * * *
* * * * * * * * *
* * * * * * * * *
```

This operation was then repeated for each of the 8 remaining sub-squares, and so on.

If we choose a different operation we get a different fractal; for example, we may clear the upper-left and lower-right corners instead of the middle. Or we may divide the square into a different number of sub-squares.

For this problem we will specify the fractal operation as an n by n grid of characters, where each character performs a particular substitution on the corresponding sub-square. The character `.` (period) indicates that the corresponding sub-square becomes completely empty. The character `!` (exclamation mark) indicates that the corresponding sub-square becomes full and is not subject to further fractal operations. The character `?` (question mark) indicates that the corresponding sub-square remains full and is subject to further fractal operations.

The fractal above would be represented by the 3 by 3 grid:

```
???
?.?
???
```

Four iterations of the fractal denoted by the 2 by 2 grid

```
.!
?.
```

would be:

```
* * * * * * * *
* * * * * * * *
* * * * * * * *
* * * * * * * *
* * * * * * * *
* * * * * * * *
* * * * * * * *
* * * * * * * *

        * * * *
        * * * *
        * * * *
        * * * *
* * * *
* * * *
* * * *
* * * *

        * * * *
        * * * *
        * * * *
        * * * *
    * *
    * *
* *
* *

        * * * *
        * * * *
        * * * *
        * * * *
    * *
    * *
  *
*
```

Your mission, should you choose to accept it (like you have any choice!) is to print out a region of an arbitrary fractal specified by an n by n grid. Note that these fractals will get way too big to fit completely into memory, so some cleverness will be required to compute only the part to be printed.

Input Specification

The first line contains a positive integer n not exceeding 100. The next n lines contain n characters each, one of `.`, `!`, `?`, defining the fractal operation to be iterated. The remainder of the input consists of queries for regions of the fractal. Each query consists of two lines: the first line gives k, the number of iterations applied to the object (assume that iteration 0 is a completely filled square of size n^k by n^k); the second line gives b, t, l, r specifying the bottom and top row and left and right column of the region to be printed. k will not exceed 100, and b, t, l, r will not exceed one million. The width and height of the region to be printed will not exceed 100. Note that rows are numbered from bottom to top (starting from 1) and columns from left to right (also starting from 1). Input is terminated by a line containing `-1`.

Output Specification

Print the region of the fractal, one line per row. Print the top row first and the bottom row last. Output a `*` for a filled-in portion and a space for a blank section. To make the output appear square, leave a single horizontal space between elements. Leave a blank line in the output after each region.

Sample Input

```
2
.!
?.
3
2 8 1 7
-1
```

Sample Output

```
        * * *
        * * *
        * * *
        * * *
    * *
    * *
  *
```