A ASM – The Abelian Sandpile Model

Problem

Modeling sandpiles is an interesting problem in statistical physics. When dropping a sand grain on an existing pile, most of the time the grain will stick to it or a couple of grains will slide down. Occasionally, however, adding one extra grain to a pile will lead to a huge avalanche of sand grains falling down.

A simple way to model sandpiles is the *Abelian Sandpile Model*. In this simple model the sandpile is described as a two-dimensional lattice with on each lattice site a height (the number of sand grains on that lattice site). When an additional grain is dropped on a lattice site, its height is increased by one. If the height becomes larger than a certain critical height, sand grains begin to topple. This is modeled by reducing the number of sand grains on the site that is too high by four, and increasing the heights of its four neighbors by one. If some of the neighbors exceed the critical height after this toppling, sand grains topple from those points too until the situation is stable again. If a sand grain falls off the lattice, the grain is gone.

Given an initial sandpile and the positions where the grains are dropped, determine the final sandpile.

Input

The first line of the input file contains a single number: the number of test cases to follow. Each test case has the following format:

- One line with four integers, y, x, n and h with $1 \leq y, x \leq 100$, $0 \leq n \leq 100$ and $3 \leq h \leq 9$: the dimensions of the sandpile, the number of dropped sand grains and the critical height.
- y lines, each with x characters in the range '`0`'...'`9`': the heights of the initial sandpile. Each height is less than or equal to the critical height.
- n lines with two integers y_i and x_i with $1 \leq y_i \leq y$ and $1 \leq x_i \leq x$: the positions where the grains are dropped.

Output

For every test case in the input file, the output should contain y lines, each with x characters in the range '`0`'...'`9`': the heights of the final sandpile.

Example

Input

```
3
3 4 1 5
1023
2344
0000
2 3
3 4 1 4
1023
2344
0000
2 3
7 7 1 9
9999999
9999999
9999999
9999999
9999999
9999999
9999999
4 4
```

Output

```
1023
2354
0000
1034
2421
0011
7999997
9799979
9979799
9996999
9979799
9799979
7999997
```