DonutsOnTheGrid

Problem Statement

Petya likes donuts. He tries to find them everywhere. For example if he is given a grid where each cell contains a '0' or '.' he will construct donuts from the cells.

To make the donuts:

1. Petya first selects a rectangle of cells of width, w, and height, h, where both are at least 3.
2. Then he removes a rectangular hole of width w-2 and height h-2 centered in the larger rectangle.
3. For the remaining cells (a closed rectangular ring) to be a valid donut, all of the cells must contain '0' (because donuts are shaped like zeros). Cells in the hole can contain anything since they are not part of the donut.

Here is an example with three overlapping donuts.

```
..........
.00000000.
.0......0.
.0.000000.
.0.0...00.
.0.0...00.
.0.000000.
.0......0.
.00000000.
..........
```

....	....	....
.00000000.	.00000000.	.00000000.
.0......0.	.0......0.	.0......0.
.0.000000.	.0.000000.	.0.000000.
.0.0...00.	.0.0...00.	.0.0...00.
.0.0...00.	.0.0...00.	.0.0...00.
.0.000000.	.0.000000.	.0.000000.
.0......0.	.0......0.	.0......0.
.00000000.	.00000000.	.00000000.
....	....	....

The grid in this problem will be pseudo-randomly generated using the following method: You are given four ints: **H** (the grid height), **W** (the grid width), **seed** and **threshold**. Let x_0=**seed** and for all i>=0 let $x_{i+1} = (x_i * 25173 + 13849)$ modulo 65536. Process the cells of the matrix in row major order (i.e., first row left to right, second row left to right, etc.). Each time you process a cell, calculate the next x_i (starting with x_1 for the upper left corner). If it is greater than or equal to **threshold**, the current cell will contain a '.', otherwise it will contain a '0'.

Notes

-The random generation of the input is only for keeping the input size small. The author's solution does not depend on any properties of the generator, and would work fast enough for any input of allowed dimensions.

Input and Constraints

First line contains an integer n (n<200), indicating number of testcase.
In each testcase, there are four integer, H , W , seed and threshold.
-**H** will be between 1 and 350, inclusive.-**W** will be between 1 and 350, inclusive.-**seed** will be between 0 and 65535, inclusive.-**threshold** will be between 0 and 65536, inclusive.

Output

For every test case, print a line of the form “Case X: Y” where X is the serial number of outpu (starting from 1). Y is number of distinct donuts.
Two donuts are considered distinct if they either have different width or height, or if the top left hand corner is in a different location (i.e., overlapping donuts are counted).
Y can be stored in 64bit ineteger.

Sample Input and output

Input:	Output:
4 5 5 222 55555 5 6 121 58000 4 5 6 50000 4 4 1 65536	Case #1: 4 Case #2: 3 Case #3: 1 Case #4: 9

Hints

Testcase 2 , the grid generated , and three distinct donuts:

```
00000.   XXX...   XXX...   ..XXX.
0.0000   X.X...   X.X...   ..X.X.
0.000.   X.X...   X.X...   ..XXX.
000.00   XXX...   X.X...   ......
000.00   ......   XXX...   ......
```