

# 11998 Rujia Liu Loves Wario Land!

*I love a game series called “Wario Land”, so I’d like to make a very difficult (indeed!!!) problem about it :) A big thank you goes to Erjin Zhou, for the idea and reference code. And a small thank you goes to Wenbin Tang, for reminding me that “Rujia Liu” also contains the letter L!*

Suppose there are  $n$  places in the very beginning of Wario Land. The land was almost deprecated, so it does not have any roads at all! You’ll be given  $m$  operations. Execute them one by one, and output the results.

1 $x y$	Wario wants to build a direct road between place $x$ and $y$ . If $x$ and $y$ are already connected (directly or indirectly), ignore this command (because Wario thinks it’s a waste of time!).
2 $x y$	Change place $x$ ’s treasure value to $v$ . This is due to newly discovered treasures, or treasures that are stolen by someone else.
3 $x y v$	Among the places along the path between $x$ and $y$ (including $x$ and $y$ ), how many of them have treasure value $\leq v$ ? Wario also needs the product of these treasure values, <i>modulo</i> $k$ (see below).

### Input

The input contains several test cases. In each test case, the first line contains three integers  $n, m, k$  ( $1 \leq n \leq 50,000, 1 \leq m \leq 100,000, 2 \leq k \leq 33333$ ). Places are numbered from 1 to  $n$ . The second line contains  $n$  integers  $V[i]$  ( $1 \leq V[i] \leq k$ ), the initial treasure values of each place. Each of the next  $m$  lines contains an operation. For each operation,  $1 \leq x, y \leq n, 1 \leq v \leq k$ . The input is terminated by end-of-file (EOF).

### Output

For each type-3 operation, output the number of places and the product of their treasure values, *modulo*  $k$ . If there is no path between  $x$  and  $y$ , or every place along the path has treasure value  $> v$ , output a single ‘0’ (rather than ‘0 0’ or ‘0 1’).

### Obfuscation

In order to prevent you from preprocessing the operations, we adopt the following obfuscation scheme:

- Each type-1 operation becomes 1  $x + d y + d$
- Each type-2 operation becomes 2  $x + d v + d$
- Each type-3 operation becomes 3  $x + d y + d v + d$

Where  $d$  is the last integer that you output, before processing this operation. If you haven’t output anything yet,  $d = 0$ .

After the obfuscation, the sample input would be:

```
4 8 39
2 3 4 5
```

```
1 1 2
3 2 3 5
1 1 3
3 2 3 5
1 25 28
3 27 28 28
3 11 12 13
3 4 5 2
```

This is the real input that your program will read.

### Sample Input

```
4 8 39
2 3 4 5
1 1 2
3 2 3 5
1 1 3
3 2 3 5
1 1 4
3 3 4 4
3 3 4 5
3 3 4 1
```

### Sample Output

```
0
3 24
2 8
3 1
0
```