E. S-Nim

Input: `standard input`
Output: `standard output`

Arthur and his sister Caroll have been playing a game called Nim for some time now. Nim is played as follows:

- The starting position has a number of heaps, all containing some, not necessarily equal, number of beads.
- The players take turns chosing a heap and removing a positive number of beads from it.
- The first player not able to make a move, loses.

Arthur and Caroll really enjoyed playing this simple game until they recently learned an easy way to always be able to find the best move:

- Xor the number of beads in the heaps in the current position (i.e. if we have 2, 4 and 7 the *xor-sum* will be 1 as 2 xor 4 xor 7 = 1).
- If the xor-sum is 0, too bad, you will lose.
- Otherwise, move such that the xor-sum becomes 0. This is always possible.

It is quite easy to convince oneself that this works. Consider these facts:

- The player that takes the last bead wins.
- After the winning player's last move the xor-sum will be 0.
- The xor-sum will change after every move.

Which means that if you make sure that the xor-sum always is 0 when you have made your move, your opponent will never be able to win, and, thus, you will win.

Understandibly it is no fun to play a game when both players know how to play perfectly (ignorance is bliss). Fourtunately, Arthur and Caroll soon came up with a similar game, S-Nim, that seemed to solve this problem. Each player is now only allowed to remove a number of beads in some predefined set S, e.g. if we have $S = \{2, 5\}$ each player is only allowed to remove 2 or 5 beads. Now it is not always possible to make the xor-sum 0 and, thus, the strategy above is useless. Or is it?

Your job is to write a program that determines if a position of S-Nim is a losing or a winning position. A position is a winning position if there is at least one move to a losing position. A position is a losing position if there are no moves to a losing position. This means, as expected, that a position with no legal moves is a losing position.

Input

Input consists of a number of test cases. For each test case: The first line contains a number k ($0 < k \leq 100$) describing the size of S, followed by k numbers s_i ($0 < s_i \leq 10000$) describing S. The second line contains a number m ($0 < m \leq 100$) describing the number of positions to evaluate. The next m lines each contain a number l ($0 < l \leq 100$) describing the number of heaps and l numbers h_i ($0 \leq h_i \leq 10000$) describing the number of beads in the heaps. The last test case is followed by a 0 on a line of its own.

Output

For each position:

- If the described position is a winning position print a `W`.
- If the described position is a losing position print an `L`.

Print a newline after each test case.

Sample Input

```
2 2 5
3
2 5 12
3 2 4 7
4 2 3 7 12
5 1 2 3 4 5
3
2 5 12
3 2 4 7
4 2 3 7 12
0
```

Sample Output

```
LWW
WWL
```