Xorset

SEASON 7 – ROUND FOUR

Recently, as everyone eventually does, Lora got quite interested in the bitwise XOR[1] operation. In order to understand it better, she wants to have a program that helps her with a few queries.

Your task is to process the queries efficiently. To answer the queries we have to support a set of numbers that allows repetitions. A brief explanation of what we mean by sets and subsets follows:

A subset of a set is obtained by choosing a few of numbers of the set, without considering the order they were chosen in. The empty set, as well as the set of all numbers in a set, are valid subsets. Two sets are different if and only if there exists an element that is present in one of them and not present in the other. Since we allow repetitions in sets, it is important to note that we differentiate elements even if they have the same value – for example, the set {1, 2, 2} contains two different subsets {1, 2}.

The XOR of a given set is considered to be the result of applying XOR to all elements of the set. For example, the XOR of the set {1, 2, 5} is equal to (1 xor 2 xor 5) = 6.

In order to help Lora you have to write a program the efficiently processes the following three types of queries, starting with an initially empty set:

- *"1 x"* – the number *x* is added to the set
- *"2 x e"* – we must find the amount of different subsets of the set that contain an **even number of elements** and have XOR equal to *x*. For e=0, this is the whole query. For e=1 we also have to provide an example of such subset.
- *"3 x e"* – we must find the amount of different subsets of the set that contain an **odd number of elements** and have XOR equal to *x*. For e=0, this is the whole query. For e=1 we also have to provide an example of such subset.

(see the sample test clarifications for more details)

Note: Since outputting long examples for queries 2,3 would slow down the program significantly, there are a few extra restrictions regarding them (see "Constraints")

Note also that we consider the empty set to be a valid set of 0 (even number) elements

Input

The first line of the input file `xorset.in` contains a single integer **N** – the amount of queries to be processed.

Each of the next N lines contains a single query in one of the formats described above.

[1] Wikipedia - XOR

Xorset

SEASON 7 – ROUND FOUR

Output

For each query of type 2 and 3 you must output on a single line in the output file `xorset.out` the amount of subsets with the required properties. **Since this number can get very big, you must print it modulo 1 000 000 007**!

If e=0 then you should not output anything else. Outputting an example subset when this is not required is considered a wrong answer!

If e=1, then on the next line you must output the amount of elements in the example subset you've found, followed by a space-separated list of those elements. **It is not necessary** to find the shortest example, but your example must not contain over 50 elements from the set. It is guaranteed that if at least on example exists then there exists an example with less than 50 elements. **If an example does not exist at all (i.e. the amount of ways was 0) then print -1 on this line.**

Constraints

$1 \leq N \leq 100\ 000$

$0 \leq$ values of the elements in any query $< 2^{30}$

$0 \leq$ number of queries with e=1 $\leq 5\ 000$

The amount of elements in each of your examples(subsets) must not exceed 50.

Time limit: 2 sec
Memory limit: 256 MB

Sample test

Input (xorset.in)	Output (xorset.out)
10 2 0 0 1 1 1 2 1 4 2 3 1 3 3 1 1 3 1 6 2 3 1 3 3 1	1 1 2 2 1 0 -1 2 4 2 3 4 6 2 1 3

Xorset

SEASON 7 – ROUND FOUR

Clarifications

The first query is sent while the set is empty. There is a single solution – the empty subset {}. We are not required to print it (printing it would be simply outputting “0” on the next line).

The next three queries add the numbers 1, 2 and 4 – making the set {1, 2, 4}. The query “2 3 1” asks for the number of subsets with even number of elements and XOR equal to 3. There is only one such subset – {1, 2}. Since e=1, we print it.

The next query asks for a subset with XOR equal to 3 again, but with odd number of elements. Such subset does not exist and we print 0. We cannot print an example, but e=1, so we output -1 on the next line.

The following queries add the numbers 3 and 6 to the set, making it {1, 2, 3, 4, 6}.

The last two queries are the same as the earlier two. This time, however, there are two subsets that satisfy the requirements of each of the queries:

With even number of elements and XOR equal to 3:

{2, 3, 4, 6} (3 = 2 xor 3 xor 4 xor 6)

{1, 2} (3 = 1 xor 2)

With odd number of elements and XOR equal to 3:

{3} (3 = 3)

{1, 4, 6} (3 = 1 xor 4 xor 6)

Since e=1 we also print an example for each query (it is irrelevant which subsets we will print as examples since they all have less than 50 elements)