Non-Interactive Nim

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Nim is a classical strategy game played by two players. There are n piles of stones, the i-th of which contains a_i stones. Players alternate in taking turns. On each turn, the player must pick a pile with a positive number of stones and remove a positive number of stones from it. The player who can't make a move loses.

The Blue Monster wanted to add an interactive Nim problem to the Ordinary & Common Problem Collection. In this problem, your program would have to play Nim against an opponent who always plays optimally. However, the Blue Monster is too lazy to learn how to create interactive problems. Therefore, you are asked to only make moves where the opponent has only one optimal move.

You are given $a_1, a_2, \ldots, a_n$ — an instance of Nim. It is guaranteed that if both players play optimally, then the first player loses. You will play as the first player, your opponent will play as the second player. Your opponent will always make optimal moves. You have to play in such a way that after each of your moves, there is only one move that your opponent can make such that you will lose if both players play optimally after that point.

Print a sequence of such moves or declare that this is impossible. Notice that since your opponent always makes optimal moves, you know exactly what moves your opponent will make. You do not have to minimize the length of the sequence.

Input

The first line contains one integer t ($1 \le t \le 5 \cdot 10^4$) — the number of test cases. t test cases follow. Each test case is described as follows.

The first line of the test case contains one integer n ($2 \le n \le 10^5$). The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^{18}$). It is guaranteed that if both players play optimally, the first player will lose.

It is guaranteed that the sum of n over all test cases doesn't exceed 10^5.

Output

For each test case, print the answer as follows.

If playing the game in a way that the opponent always has only one optimal move is impossible, print -1. Otherwise, print an integer k ($1 \le k \le 100$) on the first line — the number of moves. On each of the next k lines, print two integers p ($1 \le p \le n$) and x, signifying that you will remove x stones from the p-th pile.

It can be shown that under the constraints of this problem, if it is possible to play in a way that the opponent always has only one optimal move, then it is possible to do that and lose the game within 100 moves.

Example

standard input	standard output
2 4 4 2 7 1 4 1 1 1 1	4 3 2 1 2 3 3 4 1 -1

Note

In the first example test, the opponent is forced to make moves $(2, 2)$, $(3, 2)$, $(1, 1)$ and $(1, 1)$. The game will play out as follows:

After your move	After opponent's move
$4, 2, 5, 1$	
	$4, 0, 5, 1$
$2, 0, 5, 1$	
	$2, 0, 3, 1$
$2, 0, 0, 1$	
	$1, 0, 0, 1$
$1, 0, 0, 0$	
	$0, 0, 0, 0$

In the second example test, no matter what move you make, there will be three nonempty piles left, each with one stone. Clearly, all choices your opponent has are equivalent.