E. Keep the Sum

time limit per test: 2.5 seconds
memory limit per test: 256 megabytes

You are given an integer k and an array a of length n, where each element satisfies $0 \le a_i \le k$ for all $1 \le i \le n$. You can perform the following operation on the array:

- Choose two distinct indices i and j ($1 \le i, j \le n$ and $i \ne j$) such that $a_i + a_j = k$.
- Select an integer x satisfying $-a_j \le x \le a_i$.
- Decrease a_i by x and increase a_j by x. In other words, update $a_i := a_i - x$ and $a_j := a_j + x$.

Note that the constraints on x ensure that all elements of array a remain between 0 and k throughout the operations.

Your task is to determine whether it is possible to make the array a non-decreasing* using the above operation. If it is possible, find a sequence of at most $3n$ operations that transforms the array into a non-decreasing one.

It can be proven that if it is possible to make the array non-decreasing using the above operation, there exists a solution that uses at most $3n$ operations.

* An array $a_1, a_2, \ldots, a_n$ is said to be non-decreasing if for all $1 \le i \le n-1$, it holds that $a_i \le a_{i+1}$.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). The description of the test cases follows.

The first line of each test case contains two integers, n and k ($4 \le n \le 2 \cdot 10^5$, $1 \le k \le 10^9$) — the length of the array a and the required sum for the operation.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le k$) — the elements of array a.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output -1 if it is not possible to make the array non-decreasing using the operation.

Otherwise, output the number of operations m ($0 \le m \le 3n$). On each of the next m lines, output three integers i, j, and x representing an operation where a_i is decreased by x and a_j is increased by x.

Note that you are **not** required to minimize the number of operations. If there are multiple solutions requiring at most $3n$ operations, you may output any.

Example

input Copy

```
4
5 100
1 2 3 4 5
5 6
1 2 3 5 4
5 7
7 1 5 3 1
10 10
2 5 3 2 7 3 1 8 4 0
```

output Copy

```
0
1
4 1 1
-1
6
1 8 2
```

```
3 5 2
5 7 3
5 9 3
8 10 5
2 10 4
```

Note

In the first test case, the array is already non-decreasing, so we do not need to perform any operations.

In the second test case, we can perform an operation with $i = 4$, $j = 1$, and $x = 1$. a_4 decreases by 1 to become $5 - 1 = 4$ while a_1 increases by 1 to become $1 + 1 = 2$. After the operation, the array becomes $[2, 2, 3, 4, 4]$, which is non-decreasing.

Note that there are other ways to make the array non-decreasing, all of which would be considered correct as long as they do not use more than $3 \cdot n = 15$ operations.

In the third test case, it is not possible to make the array non-decreasing. This is because there are no distinct pairs of indices i and j where $a_i + a_j = 7$, so no operation can be done on the array.

In the fourth test case, the array is transformed as follows:

1. $[\mathbf{0}, 5, 3, 2, 7, 3, 1, \mathbf{10}, 4, 0]$
2. $[0, 5, \mathbf{1}, 2, \mathbf{9}, 3, 1, 10, 4, 0]$
3. $[0, 5, 1, 2, \mathbf{6}, 3, \mathbf{4}, 10, 4, 0]$
4. $[0, 5, 1, 2, \mathbf{3}, 3, 4, 10, \mathbf{7}, 0]$
5. $[0, 5, 1, 2, 3, 3, 4, \mathbf{5}, 7, \mathbf{5}]$
6. $[0, \mathbf{1}, 1, 2, 3, 3, 4, 5, 7, \mathbf{9}]$