Singularity

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 megabytes

In the year of 2077, problemsetting becomes simple. Robots will generate a problem by setting some random operations and then solve it. A problemsetter only has to check whether the problem is correct or not.

Here is a problem from 2077:

Given a permutation $p_1, p_2, \cdots p_n$, it is guaranteed that n is **even**. You wish to sort the permutation, using only one type of operation:

- `FakeSort(l,r)`: You have to guarantee that $r - l + 1$ is even. Let $k = r - l + 1$, then the largest $k/2$ elements and the smallest $k/2$ elements in the continuous subsequence $p_l, p_{l+1}, \cdots p_r$ will be sorted independently. That is, let $L_1, L_2, \cdots L_{k/2}$ be the indices of the largest $k/2$ numbers, and $S_1, S_2 \cdots S_{k/2}$ be the indices of the smallest $k/2$ numbers. We first sort the numbers on the indices $L_1 \sim L_{k/2}$, then sort the numbers on the indices $S_1 \sim S_{k/2}$.

Here is a concrete example; suppose the permutation is $p = \{2, 5, 7, 1, 8, 6, 4, 3\}$. If we call `FakeSort(2,7)`:

- $k = r - l + 1 = 6$. The continuous subsequence $p_l, p_{l+1}, \cdots, p_r$ is $\{5, 7, 1, 8, 6, 4\}$; we will sort the largest 3 numbers and smallest 3 numbers of this sequence independently.
- $p = \{2, 5, \mathbf{7}, 1, \mathbf{8}, \mathbf{6}, 4, 3\}$; the largest 3 numbers are bold. After sorting, they become $p = \{2, 5, \mathbf{6}, 1, \mathbf{7}, \mathbf{8}, 4, 3\}$.
- $p = \{2, \mathbf{5}, 6, \mathbf{1}, 7, 8, \mathbf{4}, 3\}$; the smallest 3 numbers are bold. After sorting, they become $p = \{2, \mathbf{1}, 6, \mathbf{4}, 7, 8, \mathbf{5}, 3\}$.

So $p = \{2, 5, 7, 1, 8, 6, 4, 3\}$ after `FakeSort(2,7)` becomes $\{2, 1, 6, 4, 7, 8, 5, 3\}$.

Please use no more than 114 operations to sort the permutation or determine it is impossible. It can be proved that if a permutation can be sorted with this operation, there is a way to use no more than 114 operations.

Input

The input contains multiple testcases. The first line of the input contains an integer T ($1 \le T \le 10^3$), the number of testcases.

For each testcase, the first line contains an **even** integer n ($4 \le n \le 10^5$), the length of the permutation.

The second line contains n integers $p_1, p_2, \cdots, p_n$ ($1 \le p_i \le n$), the permutation you need to sort. It is guaranteed that p is a permutation.

It's guaranteed that the sum of n over all testcases does not exceed 2×10^5.

Output

For each testcase, if it is impossible to sort the permutation, print -1.

Otherwise, print an integer k ($0 \le k \le 114$), denoting the number of operations used.

In the following k lines, print l, r ($1 \le l \le r \le n, r - l + 1$ is even) in each line, denoting performing `Fakesort(l,r)`.

Example

standard input	standard output
3 4 2 1 4 3 6 3 6 5 1 2 4 8 3 2 1 7 5 4 6 8	1 1 4 -1 2 4 7 1 6