Cat

Maximum time of execution: 2 seconds / test
Maximum available memory: 256 MB

The cat Motanel's favourite activity is sorting. She generally has a row of N blocks in front of her, of distinct sizes, and she tries to sort these in increasing order of size, by using a sequence of swaps. Since cat's like pairing things, N is even. Unfortunately, the dastardly rat Sobo, tries to trick Motanel, and, each time Motanel swaps the i^{th} block and the j^{th} block, Sobo will then swap the blocks on positions (N-i+1) and (N-j+1). How can Motanel optimally sort the blocks ?

More formally, you are given a permutation $P_1, P_2 \ \ldots, P_N$. An operation on indices (i, j) is defined to swap P_i and P_j, and then to swap P_{N-i+1} and P_{N-j+1}.

TASK

Given N and P, create a minimal sequence of operations that sorts P.

INPUT

There will be multiple test cases in each input file.
On the first line of the input there will be an integer that represents the number of test cases in this file.
On the first line of each test case there will be an integer that represents the value of N.
On the next line of each test case will follow N integers, that represent $P_1, P_2 \ \ldots, P_N$ respectively.

OUTPUT

Output the answers to the T test cases in order.
If there is no way of sorting the permutation, simply print $-\mathbf{1}$ on a new line.
Otherwise, begin by outputting a line that contains two integers g and L, which represent the length that you think a minimal sequence of operations has, and the length that your sequence of operations has (for partial points, these can be different). Then continue by outputting L further lines, each of which will contain two integers i and j. Such a line represents an operation on indices (i, j). Note you may choose to let $L = 0$ if you don't aim at scoring the reconstruction points, for more details see section *Scoring*. You'll get 0 points on the subtask of a test if you ever print a negative L, or make an invalid move. An invalid move (i, j) either has $i = j$ or one of the positions out of the range $[1, N]$

CONSTRAINTS

In all tests, $2 \leq N \leq 200000$, N is even and P is a permutation: any value between 1 and N appears exactly once in P

Subtask	Score	Restrictions
Subtask 1	15	$T \leq 4000$ and $N \leq 10$
Subtask 2	10	$T \leq 10^4$ and $sum\ of\ N^2$ $over\ all\ tests \leq 10^7$ $and\ P_i \leq \frac{N}{2}$ $for\ any\ 1 \leq\ i \leq \frac{N}{2}$
Subtask 3	25	$T \leq 10^4$ and $sum\ of\ N^2$ $over\ all\ tests \leq 10^7$
Subtask 4	15	$T \leq 10^4$ and $sum\ of\ N$ $over\ all\ tests \leq 3 * 10^6$ $and\ P_i \leq \frac{N}{2}$ $for\ any\ 1 \leq\ i \leq \frac{N}{2}$
Subtask 5	35	$T \leq 10^4$ and $sum\ of\ N$ $over\ all\ tests \leq 3 * 10^6$

SCORING:

Each subtask will be scored independently. The score you'll get is the score assigned to the subtask multiplied with the minimum ratio among all tests of that subtask. The ratio of a test is computed as follows:

- If there is any subtest where you haven't properly decided whether a solution exists or not, then the ratio is **0**
- If you've made throughout the whole test more than $3 * 10^6$ moves, the ratio is **0**
- If you've given a correct optimal reconstruction of moves for all subtests inside the test, the ratio for that test is **1.0** (regardless of whether your guess of optimal length was correct)
- If you've computed the correct optimal number of moves and gave a correct, solution for all subtests inside the test, the ratio for that test is **0.7**
- If you've computed the correct optimal number of moves, the ratio is **0.4**
- If you've given a correct reconstruction of moves for every subtest that was solvable, the ratio is **0.35**
- If you have at least one wrong reconstruction for a subtest, provided you've correctly distinguished between when you can sort the permutation and when you can, the ratio for the test is **0.15**

Assume these criteria are considered in this exact order and the coefficient is given by the first condition that matches the case of the current test.

EXAMPLE:

Standard Input	*Standard Output*
4 6 2 6 4 3 1 5 10 4 9 6 3 1 10 8 5 2 7 4 4 1 3 2 10 7 8 9 10 5 6 1 2 3 4	3 3 2 4 2 3 1 2 5 5 2 8 2 5 1 4 1 3 1 2 -1 2 2 2 8 1 7