XOR Sort

You are given an integer S and an array A consisting of N non-negative integers, indexed from 1. You are allowed to perform the following operation on it: choose any index *i* ($1 \le i \le N$), choose one of its neighbors j ($1 \le j \le N$, either $j = i - 1$ or $j = i + 1$) and replace A_i with ($A_i \oplus A_j$) where $\oplus$ is the bitwise XOR operation. You can see the definition of XOR at the end of the statement.
Your goal is to transform A into a sorted array:

- If $S = 1$ then the final array must be strictly increasing, i.e. $A_i < A_{i+1}$ for $1 \le i < N$
- If $S = 2$ then the final array must be non-decreasing, i.e. $A_i \le A_{i+1}$ for $1 \le i < N$

Find any sequence of operations that achieves your goal.
You aren't required to minimize the number of operations as long as their amount doesn't exceed 40000.

Input

First line contains two integers: N and S
Next line contains N integers: elements of A

Output

First line of output should contain one integer K ($0 \le K \le 40000$) - the number of operations.
Next K lines should contain two integers each, describing operations in chronological order: the first integer is an index i of the element which is being replaced and the second one is an index j of another element involved in the operation.

Constraints

- $1 \le S \le 2$
- $2 \le N \le 1000$
- $0 \le A_i < 2^{20}$

Subtasks

1. (25 points) $2 \le N \le 150$, $S = 1$, All elements of A are distinct
2. (35 points) $2 \le N \le 200$, $S = 1$, All elements of A are distinct
3. (40 points) $2 \le N \le 1000$, $S = 2$

Examples

Input	Output
5 1 3 2 8 4 1	3 1 2 4 3 5 4
5 2 4 4 2 0 1	3 3 2 4 3 5 4

First example output explanation:
[3, 2, 8, 4, 1] -> [**1**, 2, 8, 4, 1] -> [1, 2, 8, **12**, 1] -> [1, 2, 8, 12, **13**]

Second example output explanation:
[4, 4, 2, 0, 1] -> [4, 4, **6**, 0, 1] -> [4, 4, 6, **6**, 1] -> [4, 4, 6, 6, **7**]

When performing XOR operation between a and b bits the result will be 0 if a=b and 1 otherwise.
When performing bitwise XOR operation between integers a and b, XOR results will be carried out for each of the corresponding bits:
$75 \oplus 29 = 86$
$1001011 \oplus 0011101 = 1010110$

In C/C++/Java you can use the “^” operator to perform XOR.