C - Arcade Crane

Time Limit: 2.0 seconds , Memory limit: 2G

From the 2025 ICPC Northwestern Europe Regional Contest

The local arcade just installed a new game which is a take on the classic claw machine. Inside the machine, there are n plushies arranged in a row. Above this row there is a mechanical claw which is operated by coins. For each coin inserted into the machine, the claw can be used once to grab exactly three consecutive plushies out of the row and then re-insert them somewhere in the row. The remaining plushies can always be pushed around (without changing their order) to make room for the re-insertion. The goal of the game is to sort the plushies by cuteness, and once they are sorted, the machine opens up and the person who achieves this wins *all* the plushies.

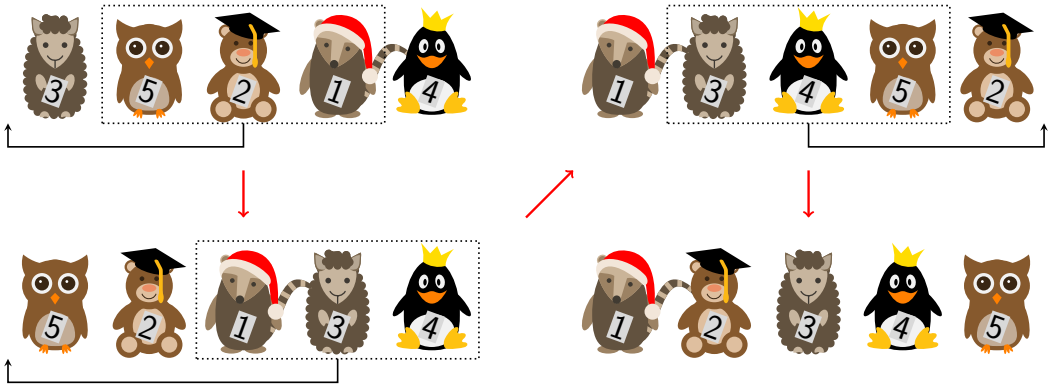

Figure 1: Illustration of Sample Output 1.

Uli really wants to win the plushies, so they have done some research and found out that each plushie has a distinct cuteness value from 1 to n. To win, they need to sort the plushies in increasing order of these values. Equipped with the knowledge of all the cuteness values and a large stash of 5000 coins, how can they operate the machine to ensure they win the plushies?

Input

The first line of input contains one integer n ($5 \leq n \leq 1000$), the number of plushies. The secone line contains n distinct integers $a_1, \ldots, a_n$ ($1 \leq a_i \leq n$ for each i), where a_i is the cuteness value of the ith plushie.

Output

The first line of output is an integer q ($0 \leq q \leq 5000$), the number of operations. Following this are q pairs of integers i and j ($1 \leq i, j \leq n-2$), describing the operations in order. Each pair of integers should be on a separate line. The plushie positions in the machine are indexed from 1 to

n. In an operation described by i and j, the plushies at positions i, $i+1$ and $i+2$ are grabbed and then re-inserted into the sequence such that they are in positions j, $j+1$ and $j+2$ after the operation.

It can be shown that a solution using at most 5000 operations always exists.

If there are multiple valid solutions, you may output any one of them. In particular, you do not need to minimize the number of operations.

Note that $i = j$ is allowed (this does not change the sequence).

Sample Input 1

```
5
3 5 2 1 4
```

Sample Output 1

```
3
2 1
3 1
2 3
```

Sample Input 2

```
6
6 5 4 3 2 1
```

Sample Output 2

```
4
1 3
2 4
3 4
1 3
```

Sample Input 3

```
9
9 2 8 5 4 6 7 3 1
```

Sample Output 3

```
5
2 6
5 1
2 7
5 3
1 3
```

Sample Input 4

```
5
1 2 3 4 5
```

Sample Output 4

```
1
2 2
```