Ranklist Sorting

You are given the scores of several players in a competition. Your task is to create a ranklist of the players, sorted in decreasing order by score.

Unfortunately, the data structure used for the list of players supports only one operation, which moves a player from position i to position j without changing the relative order of other players. If $i > j$, the positions of players at positions between j and $i-1$ increase by 1, otherwise if $i < j$ the positions of players at positions between $i+1$ and j decrease by 1.

This operation takes i steps to locate the player to be moved, and j steps to locate the position where he or she is moved to, so the overall cost of moving a player from position i to position j is $i + j$. Here, positions are numbered starting with 1.

Determine a sequence of moves to create the ranklist such that the sum of the costs of the moves is minimized.

Input

The input is read from a text file named `sorting.in` . The first line contains n ($2 \leq n \leq 1000$), the number of players. Each of the following n lines contains one non-negative integer s_i ($0 \leq s_i \leq$ 1,000,000), the scores of the players in the current order. You may assume that all scores are distinct.

Output

The output is written into a text file named `sorting.out` . In the first line of the output print the number of moves used to create the ranklist. The following lines should specify the moves in the order in which they are applied. Each move should be described by a line containing two integers i and j, which means that the player at position i is moved to position j. The numbers i and j must be separated by a single space.

Example

sorting.in	sorting.out

5
20
30
5
15
10
	2
	2 1
	3 5

Grading

30% of the test cases have values of $n \leq 10$