Monster Game (Solution)

This is a Communication Task. There is a permutation of length N, but you do not know the order of the elements. You want to sort the sequence by choosing any two elements and comparing them using a balance. However, the balance you use has a special feature. It gives the opposite result if and only if the absolute value of the difference of two chosen elements is at most 1. You may use the balance at most 25 000 times. You will get a higher score if the number of times you use the balance is smaller.

Subtask 1 ($N \leq 200$)

If $N \leq 200$, we have $\frac{200 \cdot 199}{1 \cdot 2} = 19\,900 \leq 25\,000$. Hence we can use the balance for every pair (i, j) $(0 \leq i < j \leq N - 1)$. For each element, we consider the number of times the element is judged **larger** than the other elements by the balance. For example, if $N = 6$, the results of the balance and the number of times that each element is judged as larger than the other elements are summarized as in the following table.

	0	1	2	3	4	5	The number of ‘o’
0		o	x	x	x	x	1
1	x		o	x	x	x	1
2	o	x		o	x	x	2
3	o	o	x		o	x	3
4	o	o	o	x		o	4
5	o	o	o	o	x		4

From the above table, we see that the number of times that each element is judged as **larger** than the other elements is increasing. Hence we can sort the sequence using these values. Note that the values are the same for the elements 0 and 1. We can distinguish these two elements by comparing them directly; the element judged as **smaller** is 1 and the element judged as **larger** is 0. The same is true for the elements $N - 2$ and $N - 1$.

Subtask 2 ($N \leq 1\,000$, non-adaptive grader)

Since the grader is non-adaptive for this subtask, we can use a randomized algorithm. Let us consider the quicksort.

For each recursive process, choose an arbitrary element as a pivot p. Comparing it with the other elements, we divide them into two groups — the group of elements judged as **smaller** than p and the group of elements judged as **larger** than p.

Here we recall a special feature of the balance; the element $p - 1$ is judged as **larger** than p and the element $p + 1$ is judged as **smaller** than p. Therefore, we need to find the minimum among the elements judged as **larger** than p and the maximum among the elements judged as **smaller** than p, and swap these two elements. The number of times we need to use the balance to find the minimum (or maximum) is proportional to the number of elements. Note that the difference between the minimum (or maximum) and the other elements is at least 2. We can find the minimum (or maximum) correctly using the special balance.

After we divide the elements into the two groups correctly, we can do the same for the elements in each group. We can sort the sequence recursively. Note that this algorithm does not work well if the number of elements is at most 3. To remedy this situation, if the number of elements in a group becomes at most 3, we need to choose a new pivot. Moreover, if this algorithm does not work well for every choice of a pivot (i.e., if the number of elements is at most 8), we need to sort the sequence using a simple algorithm as in Subtask 1. (If we do not consider the elements $p - 1$ and $p + 1$ as elements in the groups in the next recursive step, we need to use a simple algorithm if the number of elements becomes at most 10.)

By this algorithm, the expected value of the number of times of using the balance is $O(N \log N)$. Hence we can solve Subtask 2. However, the situation becomes worse if we need to use the balance very often. If a chosen pivot is close to the extreme values (0 or $N - 1$), we need to use the balance $O(N^2)$ times.

Subtask 3 ($N \leq 1\,000$, Slope Scoring)

We shall consider a sorting algorithm based on comparison of elements such that the number of times of using the balance is $O(N \log N)$. For example, the insertion sort using binary search, the merge sort, and the heap sort satisfy this condition. If we apply any of these sorting algorithms using the special balance, the sequence we obtain satisfies a rather special condition. Here we give an example. Let us shuffle 12 elements, and apply the merge sort to them using the special balance.

```
   7 8 11 14 0 2 6 10 1 9 4 5 3 13 15 12
-> 1 0 3 2 6 5 4 9 8 7 11 10 14 13 12 15
== [1 0] [3 2] [6 5 4] [9 8 7] [11 10] [14 13 12] [15]
```

As above, if we sort a sequence using a sorting algorithm based on comparison, the whole set of elements is divided into small groups, and the elements in each group are reversed. Thus we need to reverse the elements in each small group.

To do this, we need to specify a single element. Assume that we have specified the element 0. We shall move the element 0 to the top of the sequence and compare it with the other elements from the beginning in order. Then we will find an element which is judged as **smaller** than 0; it is the element 1. We shall reverse the elements until 1. Then we shall do the same thing using the last element obtained by reversing the elements until 1.

```
0 4 3 2 1 8 7 6 5 11 ...
^       ^

0 1 2 3 4 8 7 6 5 11 ...
        ^       ^
```

How can we specify a single element? There are several algorithms to accomplish it. The score of this task depends on the efficiency of the chosen algorithm. Here are some examples.

Algorithm 1 : Deciding a pivot

Specify a pivot between 1 and $N-2$, inclusive. Comparing it with the other elements, and divide them into the elements judged as **smaller** than the pivot and the elements judged as **larger** than the pivot. Sort the sequence using an algorithm based on comparison. Since we have specified the pivot, we shall reverse the elements using the pivot as the origin.

Algorithm 2 : Finding the element 0 among several elements from the beginning of the sequence

In a shuffled sequence, the length of a small group is about 1 to 5. We can specify the element 0 by sorting several elements (for example, 10 elements) from the beginning of the sequence using the algorithm as in Subtask 1. After specifying the element 0, we shall reverse the elements using it as the origin.

It is also possible to specify several elements from the beginning by a case-by-case analysis.

Finally, we give an estimate of the number of times the balance is used.

Note that, in a slope scoring, small improvements are also important. The score can vary if we eliminate biases in the sequence by shuffling it, or avoid using the balance more than once for a pair of elements by memorizing it.

Though the maximum number of times we need to use the balance to sort a sequence by an algorithm based on comparison is 8 977, the actual number is usually much smaller. In particular, for the merge sort, it does not exceed 8 800 with high probability.

To reverse the elements, we need to use the balance about 1 000 times.

Therefore, we will probably get full score if we can specify the origin by using the balance about 200 times.

The sample source code implementing Algorithm 2 is available on the contest website.