Ordinamento a paletta (paletta)

Time limit: 0.2 seconds
Memory limit: 256 MiB

Romeo attended a special barbecue party where the cook handled a large number of hamburgers in an amazing way. When the hamburgers needed to be flipped, the cook was able to do that on three consecutive burgers with a single spatula (paletta), quickly and in a single shot! This inspired Romeo for a new sorting problem called *paletta-sort*.

Given an array `V` storing all the integers from 0 to $N-1$ (where array positions are from 0 to $N-1$), the only feasible operation in the paletta-sort is called *ribalta*: it replaces the integers A, B, C in three consecutive positions of `V` with their flipped values C, B, A in this order. You are required to help Romeo to understand if paletta-sort can sort `V`: if it is so, say how many *ribalta* operations are needed.

Implementation

You shall submit one file having extension `.c`, `.cpp` or `.pas`.

> ☞ Among the attachments of this task you will find a template (`paletta.c`, `paletta.cpp`, `paletta.pas`) with a sample incomplete implementation.

You need to implement the following function:

C/C++	`long long paletta_sort(int N, int V[]);`
Pascal	`function paletta_sort(N: longint; V: array of longint) : int64;`

- N is an integer representing the number of elements to sort.
- `V` is an array, indexed from 0 to $N-1$, containing the elements to sort.
- The function has to return the number of *ribalta* operations to sort `V`, or -1 if the latter cannot be sorted in this way.

The grader will call the function `paletta_sort` and will print the returned value to the output file.

Grader

In the directory for this problem there is a simplified version of the grader used during evaluation, which you can use to test your solutions locally. The sample grader reads data from `stdin`, calls the function that you should implement and writes to `stdout` in the following format.

The input file is made of 2 lines, containing:

- Line 1: integer N.
- Lines 2: values `V[`i`]` for $i = 0, \ldots, N-1$..

The output file is made of a single line, containing:

- Line 1: the value returned by the function `paletta_sort`.

Constraints

- $1 \leq N \leq 1\,500\,000$.
- $0 \leq \texttt{V}[i] \leq N - 1$ for $i = 0, \ldots, N - 1$.

Scoring

Your program will be tested against several test cases grouped in subtasks. For each test case you will get the following factor.

- 1: If you compute the minimal number of *ribalta* operations.
- 0.2: If array `V` can be sorted and you compute any non-negative number (that is, you can distinguish if `V` can be sorted or not.
- 0: All the remaning cases.

For each subtask, its score is given by the product of its weight below times the above factor for the worst test case in the subtask.

- **Subtask 1** [**5 score**]: Examples.
- **Subtask 2** [**19 score**]: $N \leq 100$.
- **Subtask 3** [**24 score**]: $N \leq 5000$.
- **Subtask 4** [**21 score**]: $R \leq 100$ (or `V` cannot be sorted).
- **Subtask 5** [**25 score**]: $N \leq 100\,000$.
- **Subtask 6** [**6 score**]: No limitations.

Examples

input.txt	output.txt
5 2 0 4 3 1	-1
6 2 3 0 5 4 1	3

Explanation

In the **first example** it is not possible to sort `V`.

In the **first example**, the proposed solution yields the following sequence of *ribalta* operations:

2	3	0	5	4	1

2	3	0	1	4	5

0	3	2	1	4	5

0	1	2	3	4	5