Senate Building Decoration (decoration)

You are involved in building a grand new Senate building, and a key task is to choose its central decoration. The Senate wants the most expensive decoration possible. However, the Plebeians also wish to be part of the selection, and they are concerned that a very expensive decoration will result in higher taxes. Thus, they want to select the cheapest decoration.

To decide, a row of N potential decorations is presented, arranged in a line. Each decoration, indexed from 0 to $N-1$, has a specific cost, denoted by A_i.

After lengthy discussions, the following decision method has been chosen. The Senate and the Plebeians take turns making a choice. **The Senate plays first.** Their goals are opposed:

- The Senate aims to ensure the final decoration has the **highest** possible cost.
- The Plebeians aim for the final decoration to have the **lowest** possible cost.

On each turn, the current player (Senate or Plebeians) considers the decorations that are still available. They must select a contiguous block (a subarray) of these decorations; the selected block must **not be the entire current block** and must be **at least half** (rounding up) the length of the current block of decorations. All decorations lying **outside** this chosen block are then removed.

This process continues, with the row of decorations shrinking after each turn, until only a single decoration remains. The cost of this final decoration is the result. Your task is to predict this cost, assuming both the Senate and the Plebeians play optimally to achieve their respective goals.

Implementation

You will have to submit a single `.cpp` source file.

> ⚠ Among this task's attachments you will find a template `decoration.cpp` with a sample implementation.

You will have to implement the following function:

C++	`int decorate(int N, vector<int> A)`

- Integer N represents the initial number of potential decorations.
- The array A, indexed from 0 to $N-1$, contains the values $A_0, A_1, ..., A_{N-1}$, where A_i is the cost of the i-th decoration.
- The function should return the cost of the selected decoration.

Sample Grader

Among this task's attachments you will find 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 `decorate` and writes back on `stdout` using the following format.

The input file is formed by 2 lines, containing:

- Line 1: the integer N.
- Line 2: N space-separated integers, representing the elements $A_0, A_1, ..., A_{N-1}$.

The output is made up of a single line, containing the value returned by the function `decorate`.

Constraints

- $1 \leq N \leq 5000$.
- $0 \leq A_i \leq 1\,000\,000\,000$.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

- **Subtask 0 [0 points]**: Sample test cases.
- **Subtask 1 [12 points]**: $N \leq 1500$, A is non-decreasing (i.e. for $i < j$, $A_i \leq A_j$).
- **Subtask 2 [4 points]**: $N \leq 10$, $A_i \leq 1000$
- **Subtask 3 [22 points]**: $N \leq 250$
- **Subtask 4 [26 points]**: $N \leq 800$
- **Subtask 5 [33 points]**: $N \leq 1500$
- **Subtask 6 [3 points]**: No additional constraints.

Examples

stdin	stdout
5 0 1 2 3 4	3
5 4 3 1 1 5	3

Explanation

In the first sample case, it can be shown that the following sequence of moves is optimal:

- Senate's first move: The only subarray, that maximizes the worst possible outcome for them: $[2, 3, 4]$.
- Plebeians' reply: From $[2, 3, 4]$ they have to choose subarray of length at least two; choosing $[2, 3]$ minimises the eventual result.
- Finally the Senate chooses 3.

In the second example:

- Senate's first move: The only opening that keeps their guaranteed value above 2 is the block $[4, 3, 1]$ (length 3, at least half of 5 and not the whole row).
- Plebeians' reply: From $[4, 3, 1]$ they must keep a block of length 2; choosing $[3, 1]$ minimises the eventual result.
- Finally the Senate chooses 3.

Any other first move (e.g. $[3, 1, 1]$, $[1, 1, 5]$, or a length-4 block) lets the Plebeians force the outcome down to 1, so optimal play from both sides ends with the cost 3.