Task 2: Discharging (`discharging`)

In a quest to be the very best, Pichuu the electric mouse has since started a new business that better caters to his strengths: using his favorite move Discharge to charge his customers' phones. A super effective business, it enjoys the patronage of many customers daily.

On a particular day, Pichuu has N customers queuing in wait to charge their phones. He is able to charge multiple phones simultaneously, where the power provided to each phone is equal and constant. Of course, different phone models possess different battery capacities, thus requiring varying lengths of time to charge fully. More specifically, the i^{th} phone takes T_i minutes to become fully charged.

When discharging, Pichuu does not stop until all the phones are fully charged. As such, to avoid getting shocked, customers are forced to wait until the last phone has finished charging before they retrieve their phones. However not all hope is lost: to minimise the total amount of time his customers spend waiting, Pichuu can divide them into an arbitrary amount of contiguous groups, before charging the groups in order. Thus, each group has to wait for the groups in front of them to finish before Pichuu can begin charging their phones.

Every customer will belong to exactly one group, where the i^{th} customer is assigned to the G_i^{th} group. Let the maximum T_i in the k^{th} group be M_k. Hence, the total amount of time spent waiting by the i^{th} customer, W_i, would be the sum of time taken for each group before him as well as his own group. **(For an illustration, refer to explanation for sample testcases)**

$$W_i = \sum_{n=1}^{G_i} M_n$$

To maximise customer satisfaction, Pichuu would like to minimise the sum of the waiting times.

Pichuu has but a mouse-sized brain that is incapable of calculating the optimal way to group his customers. Therefore, he requires your help to find the optimal grouping and hence the minimum sum of waiting time.

Input

Your program must read from standard input.

The first line contains an integer N, the number of customers.

The second line contains N integers, where the i^{th} integer represents the time taken to charge the i^{th} customer's phone T_i.

Output

Your program must print to standard output.

The output should contain a single integer on a single line, the minimum possible total waiting time.

Implementation Note

As the input lengths for subtasks 3, 4 and 6 may be very large, you are recommended to use C++ with fast input routines to solve this problem. The scientific committee does not have a solution written in Python that can fully solve this problem.

C++ and Java source files containing fast input/output templates have been provided in the attachment. You are strongly recommended to use these templates.

If you are implementing your solution in Java, please name your file `Discharging.java` and place your main function inside `class Discharging`.

Subtasks

The maximum execution time on each instance is 1.0s, and the maximum memory usage on each instance is 1GiB. For all testcases, the input will satisfy the following bounds:

- $1 \leq N \leq 10^6$
- $1 \leq T_i \leq 10^9$

Your program will be tested on input instances that satisfy the following restrictions:

Subtask	Marks	Additional Constraints
1	9	$1 \leq N \leq 3$
2	13	$1 \leq N \leq 1500$ T_i is in non-decreasing order.
3	25	T_i is in non-decreasing order.
4	11	T_i is in non-increasing order.
5	14	$1 \leq N \leq 1500$
6	28	-

Sample Testcase 1

This testcase is valid for subtasks 5 and 7.

Input	Output
5 1 3 2 6 3	27

Sample Testcase 1 Explanation

It is optimal to split the customers into two contiguous groups of (1, 3, 2) and (6, 3). The time taken for the two groups are 3 and 6 respectively as they are the maximum T_i in the groups. The waiting time for the first group is 3 per customer while the waiting time for the second group is 3 + 6 = 9 per customers as the second group has to wait for the first group to finish. Thus the total waiting time is 3 + 3 + 3 + 9 + 9 = 27.

Sample Testcase 2

This testcase is valid for subtasks 2, 3, 5 and 6.

Input	Output
7 1 1 2 2 2 2 2	14

Sample Testcase 2 Explanation

The optimal grouping is to simply have all the customers in one group and charge their phones all at once. Thus the time taken for this group would be 2. The total waiting time would then be 2 + 2 + 2 + 2 + 2 + 2 + 2 = 14.