Problem A. Optimizing Program

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

The ICPC finals will be held soon, so Yergeldi and his team needs your help. While they were preparing for the competition, they faced an interesting task. You have a list of length N which consists of arrays of different lengths. You have one single operation, you can take any two arrays and merge them into one, the cost of the operation is equal to the sum of their lengths. As a result, you will have a list of N - 1 arrays. The process repeats until there is only one final array left. Find out for what minimum cost it is possible to combine all arrays.

Input

The first line contains an integer n ($1 \leqslant n \leqslant 2 \cdot 10^5$), the size of the list a. The next line contains n positive integers a_1, a_2, ..., a_n ($1 \leqslant a_i \leqslant 2 \cdot 10^5$), representing the sizes of arrays in the list a.

Output

Print a single integer - the minimum cost of operations.

Examples

standard input	standard output
4 6 5 3 9	45
10 42 18 63 26 19 15 11 29 26 24	869

Note

Explanation for the first test case:

[6, 5, 3, 9] –> First, merge arrays of lengths 5 and 3 that will cost 8.

[6, 8, 9] –> Next, merge arrys of lengths 6 and 8 that will cost 14.

[14, 9] –> Finally, merge the remaining two arrays that will cost 23.

Therefore, the total cost for merging all arrays is $8 + 14 + 23 = 45$.

Problem B. Rock Game

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

We have a collection of rocks, each rock has a positive integer weight.

With each turn, we choose the two **heaviest** rocks and smash them together. Suppose the stones have weights x and y with $x \leqslant y$. If $x = y$, both stones are totally destroyed. If $x \neq y$, the stone of weight x is totally destroyed, and the stone of weight y has a new weight $y - x$.

At the end, there is at most 1 stone left. Output the weight of this stone (or 0 if there are no stones left).

Input

The first line contains integer N $(1 \leqslant N \leqslant 10^5)$ - the number of stones.

The second line contains N integers a_i $(1 \leqslant a_i \leqslant 10^9)$ - the weights of each stone.

Output

Print the weight of the remaining stone, or 0 if no one left.

Example

standard input	standard output
6 2 7 4 1 8 1	1

Note

We combine 7 and 8 to get 1 so the array converts to [2,4,1,1,1] then, we combine 2 and 4 to get 2 so the array converts to [2,1,1,1] then, we combine 2 and 1 to get 1 so the array converts to [1,1,1] then, we combine 1 and 1 to get 0 so the array converts to [1] then that's the value of last stone.

Problem C. Standard problem about soccer

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Kakyoin loves football and he goes to the final of the World Cup. At the stadium, he noticed that there are n rows, each can accommodate a distinct number of people. The price of the ticket depends on the row. If there are k ($k > 0$) free seats in the row, then the price of one ticket will be equal to k. What is the maximum amount of money stadium management can get if there are x people in line for a ticket?

Input

The first line consists of n and x ($1 \leqslant n, x \leqslant 10^5$). n denotes the number of seating rows in the stadium and x denotes the number of football fans waiting in line to get a ticket for the match.

Next line consists of n space separated integers a_1, a_2, a_3, ..., a_n where a_i ($1 \leqslant a_i \leqslant 10^5$) denotes the number of empty seats initially in the i-th row.

It is guaranteed that there are enough free seats for all visitors.

Output

Print one integer - the maximum amount of money the stadium can earn.

Examples

standard input	standard output

`3 10`
`6 8 9`
	`67`

`1 2`
`5`
	`9`

Problem D. Experiment with Mixtures

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

Mark is making an experiment with mixtures of different densities. For his experiment he wants all of the mixtures to have a density $\geqslant m$.

He uses the following formula to obtain a combined mixture: $d_{new} = d_{least} + 2 \cdot d_{second\ least}$, where d_{new} is the density of the new mixture, d_{least} is the smallest density among all mixtures and $d_{second\ least}$ is the second smallest density among all mixtures.

Mark repeats the mixing until he gets all the mixtures with the density $\geqslant m$.

You are given the densities of mixtures. How many times Mark should mix his mixtures to get the densities of all mixtures $\geqslant m$?

Input

The first line consists of integers n and m ($1 \leqslant n \leqslant 10^6$, $0 \leqslant m \leqslant 10^9$), the number of mixtures and the minimum required density correspondingly.

The next line contains n space-separated integers d_i ($0 \leqslant d_i \leqslant 10^6$) describing the densities of mixtures.

Output

Print the number of operations that are needed to make all the densities $\geqslant m$. If it is impossible, print -1.

Examples

standard input	standard output

`3 10`
`1 1 1`
	`-1`

`6 7`
`1 2 3 9 10 12`
	`2`

Problem E. K-th sum

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Nurdana loves cookies! She stores all the cookies in separate boxes, but mom allows her to store only k boxes. Her dad is an assistant in this matter. He gives her a box in which there are n cookies. Help her calculate how many cookies she can have?

Input

The first line contains two integers q and k ($1 \le q, k \le 10^5$). Each of the following q lines contains one command.

There are two types of commands:

- *insert* n - dad gives a box with n cookies ($0 \leqslant n \leqslant 10^9$)
- *print* - print the maximum number of cookies Nurdana can have

Output

For each query of type *print* print the maximum sum of cookies if consider no more than k boxes.

Example

standard input	standard output

`6 4`
`print`
`insert 9`
`insert 6`
`print`
`insert 10`
`print`
	`0`
	`15`
	`25`

Note

This problem must be solved using a heap.

Use long long type for the sum of elements.

Since k is fixed, you don't need to store all numbers, only k largest elements.