Codeforces Round #538 (Div. 2)

A. Got Any Grapes?

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

The Duck song

For simplicity, we'll assume that there are only three types of grapes: green grapes, purple grapes and black grapes.

Andrew, Dmitry and Michal are all grapes' lovers, however their preferences of grapes are different. To make all of them happy, the following should happen:

- Andrew, Dmitry and Michal should eat at least x, y and z grapes, respectively.
- Andrew has an extreme affinity for green grapes, thus he will eat green grapes and green grapes only.
- On the other hand, Dmitry is not a fan of black grapes — any types of grapes except black would do for him. In other words, Dmitry can eat green and purple grapes.
- Michal has a common taste — he enjoys grapes in general and will be pleased with any types of grapes, as long as the quantity is sufficient.

Knowing that his friends are so fond of grapes, Aki decided to host a grape party with them. He has prepared a box with a green grapes, b purple grapes and c black grapes.

However, Aki isn't sure if the box he prepared contains enough grapes to make everyone happy. Can you please find out whether it's possible to distribute grapes so that everyone is happy or Aki has to buy some more grapes?

It is not required to distribute all the grapes, so it's possible that some of them will remain unused.

Input

The first line contains three integers x, y and z ($1 \le x, y, z \le 10^5$) — the number of grapes Andrew, Dmitry and Michal want to eat.

The second line contains three integers a, b, c ($1 \le a, b, c \le 10^5$) — the number of green, purple and black grapes in the box.

Output

If there is a grape distribution that allows everyone to be happy, print "YES", otherwise print "NO".

Examples

input
1 6 2 4 3 3
output
YES

input
5 1 1 4 3 2
output
NO

Note

In the first example, there is only one possible distribution:

Andrew should take 1 green grape, Dmitry should take 3 remaining green grapes and 3 purple grapes, and Michal will take 2 out of 3 available black grapes.

In the second test, there is no possible distribution, since Andrew is not be able to eat enough green grapes. :(

B. Yet Another Array Partitioning Task

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

An array b is called to be a *subarray* of a if it forms a continuous subsequence of a, that is, if it is equal to a_l, a_{l+1}, ..., a_r for some l, r.

Suppose m is some known constant. For any array, having m or more elements, let's define it's *beauty* as the sum of m largest elements of that array. For example:

- For array $x = [4, 3, 1, 5, 2]$ and $m = 3$, the 3 largest elements of x are 5, 4 and 3, so the beauty of x is $5 + 4 + 3 = 12$.
- For array $x = [10, 10, 10]$ and $m = 2$, the beauty of x is $10 + 10 = 20$.

You are given an array $a_1, a_2, \ldots, a_n$, the value of the said constant m and an integer k. Your need to split the array a into exactly k subarrays such that:

- Each element from a belongs to exactly one subarray.
- Each subarray has at least m elements.
- The sum of all beauties of k subarrays is maximum possible.

Input

The first line contains three integers n, m and k ($2 \le n \le 2 \cdot 10^5$, $1 \le m$, $2 \le k$, $m \cdot k \le n$) — the number of elements in a, the constant m in the definition of beauty and the number of subarrays to split to.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($-10^9 \le a_i \le 10^9$).

Output

In the first line, print the maximum possible sum of the beauties of the subarrays in the optimal partition.

In the second line, print $k - 1$ integers $p_1, p_2, \ldots, p_{k-1}$ ($1 \le p_1 < p_2 < \ldots < p_{k-1} < n$) representing the partition of the array, in which:

- All elements with indices from 1 to p_1 belong to the first subarray.
- All elements with indices from $p_1 + 1$ to p_2 belong to the second subarray.
- ….
- All elements with indices from $p_{k-1} + 1$ to n belong to the last, k-th subarray.

If there are several optimal partitions, print any of them.

Examples

input
9 2 3 5 2 5 2 4 1 1 3 2
output
21 3 5

input
6 1 4 4 1 3 2 2 3
output
12 1 3 5

input
2 1 2 -1000000000 1000000000
output
0 1

Note

In the first example, one of the optimal partitions is $[5, 2, 5]$, $[2, 4]$, $[1, 1, 3, 2]$.

- The beauty of the subarray $[5, 2, 5]$ is $5 + 5 = 10$.
- The beauty of the subarray $[2, 4]$ is $2 + 4 = 6$.
- The beauty of the subarray $[1, 1, 3, 2]$ is $3 + 2 = 5$.

The sum of their beauties is $10 + 6 + 5 = 21$.

In the second example, one optimal partition is $[4]$, $[1, 3]$, $[2, 2]$, $[3]$.

C. Trailing Loves (or L'oeufs?)

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

The number "zero" is called "love" (or

"l'oeuf" to be precise, literally means "egg" in French), for example when denoting the zero score in a game of tennis.

Aki is fond of numbers, especially those with trailing zeros. For example, the number 9200 has two trailing zeros. Aki thinks the more trailing zero digits a number has, the prettier it is.

However, Aki believes, that the number of trailing zeros of a number is not static, but depends on the base (radix) it is represented in. Thus, he considers a few scenarios with some numbers and bases. And now, since the numbers he used become quite bizarre, he asks you to help him to calculate the beauty of these numbers.

Given two integers n and b (in decimal notation), your task is to calculate the number of trailing zero digits in the b-ary (in the base/radix of b) representation of $n\,!$ (factorial of n).

Input
The only line of the input contains two integers n and b ($1 \le n \le 10^{18}$, $2 \le b \le 10^{12}$).

Output
Print an only integer — the number of trailing zero digits in the b-ary representation of $n!$

Examples

input
6 9
output
1

input
38 11
output
3

input
5 2
output
3

input
5 10
output
1

Note
In the first example, $6!_{(10)} = 720_{(10)} = 880_{(9)}$.

In the third and fourth example, $5!_{(10)} = 120_{(10)} = 1111000_{(2)}$.

The representation of the number x in the b-ary base is $d_1, d_2, \ldots, d_k$ if $x = d_1 b^{k-1} + d_2 b^{k-2} + \ldots + d_k b^0$, where d_i are integers and $0 \le d_i \le b - 1$. For example, the number 720 from the first example is represented as $880_{(9)}$ since $720 = 8 \cdot 9^2 + 8 \cdot 9 + 0 \cdot 1$.

You can read more about bases here.

D. Flood Fill

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given a line of n colored squares in a row, numbered from 1 to n from left to right. The i-th square initially has the color c_i.

Let's say, that two squares i and j belong to the same connected component if $c_i = c_j$, and $c_i = c_k$ for all k satisfying $i < k < j$. In other words, all squares on the segment from i to j should have the same color.

For example, the line $[3, 3, 3]$ has 1 connected component, while the line $[5, 2, 4, 4]$ has 3 connected components.

The game "flood fill" is played on the given line as follows:

- At the start of the game you pick any starting square (this is not counted as a turn).
- Then, in each game turn, change the color of the connected component containing the starting square to any other color.

Find the minimum number of turns needed for the entire line to be changed into a single color.

Input
The first line contains a single integer n ($1 \le n \le 5000$) — the number of squares.

The second line contains integers $c_1, c_2, \ldots, c_n$ ($1 \le c_i \le 5000$) — the initial colors of the squares.

Output
Print a single integer — the minimum number of the turns needed.

Examples

input
4 5 2 2 1
output
2

input
8 4 5 2 2 1 3 5 5
output
4

input
1 4
output
0

Note
In the first example, a possible way to achieve an optimal answer is to pick square with index 2 as the starting square and then play as follows:

- $[5, 2, 2, 1]$
- $[5, 5, 5, 1]$
- $[1, 1, 1, 1]$

In the second example, a possible way to achieve an optimal answer is to pick square with index 5 as the starting square and then perform recoloring into colors $2, 3, 5, 4$ in that order.

In the third example, the line already consists of one color only.

E. Arithmetic Progression

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

This is an interactive problem!

An arithmetic progression or arithmetic sequence is a sequence of integers such that the subtraction of element with its previous element ($x_i - x_{i-1}$, where $i \ge 2$) is constant — such difference is called a *common difference* of the sequence.

That is, an arithmetic progression is a sequence of form $x_i = x_1 + (i - 1)d$, where d is a common difference of the sequence.

There is a secret list of n integers $a_1, a_2, \ldots, a_n$.

It is guaranteed that all elements $a_1, a_2, \ldots, a_n$ are between 0 and 10^9, inclusive.

This list is special: if sorted in increasing order, it will form an arithmetic progression with positive common difference ($d > 0$). For example, the list $[14, 24, 9, 19]$ satisfies this requirement, after sorting it makes a list $[9, 14, 19, 24]$, which can be produced as $x_n = 9 + 5 \cdot (n - 1)$.

Also you are also given a device, which has a quite discharged battery, thus you can only use it to perform at most 60 queries of following two types:

- Given a value i ($1 \le i \le n$), the device will show the value of the a_i.
- Given a value x ($0 \le x \le 10^9$), the device will return 1 if an element with a value strictly greater than x exists, and it will return 0 otherwise.

Your can use this special device for at most 60 queries. Could you please find out the smallest element and the common difference of the sequence? That is, values x_1 and d in the definition of the arithmetic progression. Note that the array a is not sorted.

Interaction

The interaction starts with a single integer n ($2 \le n \le 10^6$), the size of the list of integers.

Then you can make queries of two types:

- "? i" ($1 \le i \le n$) — to get the value of a_i.
- "> x" ($0 \le x \le 10^9$) — to check whether there exists an element greater than x

After the query read its result r as an integer.

- For the first query type, the r satisfies $0 \le r \le 10^9$.
- For the second query type, the r is either 0 or 1.
- In case you make more than 60 queries or violated the number range in the queries, you will get a $r = -1$.
- If you terminate after receiving the -1, you will get the "Wrong answer" verdict. Otherwise you can get an arbitrary verdict because your solution will continue to read from a closed stream.

When you find out what the smallest element x_1 and common difference d, print

- "! x_1 d"

And quit after that. This query is not counted towards the 60 queries limit.

After printing any query do not forget to output end of line and flush the output. Otherwise you will get Idleness limit exceeded. To do this, use:

- fflush(stdout) or cout.flush() in C++;
- System.out.flush() in Java;
- flush(output) in Pascal;
- stdout.flush() in Python;
- see documentation for other languages.

Hacks

For hack, use the following format:

The first line should contain an integer n ($2 \le n \le 10^6$) — the list's size.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 10^9$) — the elements of the list.

Also, after the sorting the list must form an arithmetic progression with positive common difference.

Example

input
4 0 1 14 24 9 19
output
> 25 > 15 ? 1 ? 2 ? 3 ? 4 ! 9 5

Note

Note that the example interaction contains extra empty lines so that it's easier to read. The real interaction doesn't contain any empty lines and you shouldn't print any extra empty lines as well.

The list in the example test is $[14, 24, 9, 19]$.

F. Please, another Queries on Array?

time limit per test: 5.5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given an array $a_1, a_2, \ldots, a_n$.

You need to perform q queries of the following two types:

1. "MULTIPLY l r x" — for every i ($l \le i \le r$) multiply a_i by x.
2. "TOTIENT l r" — print $\varphi(\prod_{i=l}^{r} a_i)$ taken modulo $10^9 + 7$, where φ denotes Euler's totient function.

The Euler's totient function of a positive integer n (denoted as $\varphi(n)$) is the number of integers x ($1 \le x \le n$) such that $\gcd(n, x) = 1$.

Input

The first line contains two integers n and q ($1 \le n \le 4 \cdot 10^5$, $1 \le q \le 2 \cdot 10^5$) — the number of elements in array a and the number of queries.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 300$) — the elements of array a.

Then q lines follow, describing queries in the format given in the statement.

1. "MULTIPLY l r x" ($1 \le l \le r \le n$, $1 \le x \le 300$) — denotes a multiplication query.
2. "TOTIENT l r" ($1 \le l \le r \le n$) — denotes a query on the value of Euler's totient function.

It is guaranteed that there is at least one "TOTIENT" query.

Output

For each "TOTIENT" query, print the answer to it.

Example

input
4 4 5 9 1 2 TOTIENT 3 3 TOTIENT 3 4 MULTIPLY 4 4 3 TOTIENT 4 4
output
1 1 2

Note

In the first example, $\varphi(1) = 1$ for the first query, $\varphi(2) = 1$ for the second query and $\varphi(6) = 2$ for the third one.