Kotlin Heroes: Practice 6

A. Remove Duplicates

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

Petya has an array a consisting of n integers. He wants to remove duplicate (equal) elements.

Petya wants to leave only the rightmost entry (occurrence) for each element of the array. The relative order of the remaining unique elements should not be changed.

Input
The first line contains a single integer n ($1 \le n \le 50$) — the number of elements in Petya's array.

The following line contains a sequence $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 1\,000$) — the Petya's array.

Output
In the first line print integer x — the number of elements which will be left in Petya's array after he removed the duplicates.

In the second line print x integers separated with a space — Petya's array after he removed the duplicates. For each unique element only the rightmost entry should be left.

Examples

input
6 1 5 5 1 6 1
output
3 5 6 1

input
5 2 4 2 4 4
output
2 2 4

input
5 6 6 6 6 6
output
1 6

Note
In the first example you should remove two integers 1, which are in the positions 1 and 4. Also you should remove the integer 5, which is in the position 2.

In the second example you should remove integer 2, which is in the position 1, and two integers 4, which are in the positions 2 and 4.

In the third example you should remove four integers 6, which are in the positions 1, 2, 3 and 4.

B. Water Buying

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

Polycarp wants to cook a soup. To do it, he needs to buy exactly n liters of water.

There are only two types of water bottles in the nearby shop — 1-liter bottles and 2-liter bottles. There are infinitely many bottles of these two types in the shop.

The bottle of the first type costs a burles and the bottle of the second type costs b burles correspondingly.

Polycarp wants to spend as few money as possible. Your task is to find the minimum amount of money (in burles) Polycarp needs to buy exactly n liters of water in the nearby shop if the bottle of the first type costs a burles and the bottle of the second type costs b burles.

You also have to answer q independent queries.

Input
The first line of the input contains one integer q ($1 \le q \le 500$) — the number of queries.

The next q lines contain queries. The i-th query is given as three space-separated integers n_i, a_i and b_i ($1 \le n_i \le 10^{12}, 1 \le a_i, b_i \le 1000$) — how many liters Polycarp needs in the i-th query, the cost (in burles) of the bottle of the first type in the i-th query and the cost (in burles) of the bottle of the second type in the i-th query, respectively.

Output
Print q integers. The i-th integer should be equal to the minimum amount of money (in burles) Polycarp needs to buy exactly n_i liters of water in the nearby shop if the bottle of the first type costs a_i burles and the bottle of the second type costs b_i burles.

Example

input
4 10 1 3 7 3 2 1 1000 1 1000000000000 42 88
output
10 9 1000 42000000000000

C. File Name

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

You can not just take the file and send it. When Polycarp trying to send a file in the social network "Codehorses", he encountered an unexpected problem. If the name of the file contains three or more "x" (lowercase Latin letters "x") in a row, the system considers that the file content does not correspond to the social network topic. In this case, the file is not sent and an error message is displayed.

Determine the minimum number of characters to remove from the file name so after that the name does not contain "xxx" as a substring. Print `0` if the file name does not initially contain a forbidden substring "xxx".

You can delete characters in arbitrary positions (not necessarily consecutive). If you delete a character, then the length of a string is reduced by 1. For example, if you delete the character in the position 2 from the string "exxxii", then the resulting string is "exxii".

Input
The first line contains integer n ($3 \le n \le 100$) — the length of the file name.

The second line contains a string of length n consisting of lowercase Latin letters only — the file name.

Output
Print the minimum number of characters to remove from the file name so after that the name does not contain "xxx" as a substring. If initially the file name dost not contain a forbidden substring "xxx", print `0`.

Examples

input
6 xxxiii
output
1

input
5 xxoxx
output
0

input

10 xxxxxxxxxx
output
8

Note
In the first example Polycarp tried to send a file with name contains number 33, written in Roman numerals. But he can not just send the file, because it name contains three letters "x" in a row. To send the file he needs to remove any one of this letters.

D. Substrings Sort

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

You are given n strings. Each string consists of lowercase English letters. Rearrange (reorder) the given strings in such a way that for every string, all strings that are placed before it are its *substrings*.

String a is a *substring* of string b if it is possible to choose several *consecutive* letters in b in such a way that they form a. For example, string "for" is contained as a *substring* in strings "codeforces", "for" and "therefore", but is not contained as a *substring* in strings "four", "fofo" and "rof".

Input
The first line contains an integer n ($1 \le n \le 100$) — the number of strings.

The next n lines contain the given strings. The number of letters in each string is from 1 to 100, inclusive. Each string consists of lowercase English letters.

Some strings might be equal.

Output
If it is impossible to reorder n given strings in required order, print "NO" (without quotes).

Otherwise print "YES" (without quotes) and n given strings in required order.

Examples

input
5 a aba abacaba ba aba
output
YES a ba aba aba abacaba

input
5 a abacaba ba aba abab
output
NO

input
3 qwerty qwerty qwerty
output
YES qwerty qwerty qwerty

Note
In the second example you cannot reorder the strings because the string "abab" is not a *substring* of the string "abacaba".

E. Books Queries

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

You have got a shelf and want to put some books on it.

You are given q queries of three types:

1. L id — put a book having index id on the shelf to the left from the leftmost existing book;
2. R id — put a book having index id on the shelf to the right from the rightmost existing book;
3. ? id — calculate the minimum number of books you need to pop from the left or from the right in such a way that the book with index id will be leftmost or rightmost.

You can assume that the first book you will put can have any position (it does not matter) and queries of type 3 are always valid (it is guaranteed that the book in each such query is already placed). You can also assume that you don't put the same book on the shelf twice, so ids don't repeat in queries of first two types.

Your problem is to answer all the queries of type 3 in order they appear in the input.

Note that after answering the query of type 3 all the books remain on the shelf and the relative order of books does not change.

If you are Python programmer, consider using PyPy instead of Python when you submit your code.

Input

The first line of the input contains one integer q ($1 \le q \le 2 \cdot 10^5$) — the number of queries.

Then q lines follow. The i-th line contains the i-th query in format as in the problem statement. It is guaranteed that queries are always valid (for query type 3, it is guaranteed that the book in each such query is already placed, and for other types, it is guaranteed that the book was not placed before).

It is guaranteed that there is at least one query of type 3 in the input.

In each query the constraint $1 \le id \le 2 \cdot 10^5$ is met.

Output

Print answers to queries of the type 3 in order they appear in the input.

Examples

input
8 L 1 R 2 R 3 ? 2 L 4 ? 1 L 5 ? 1
output
1 1 2

input
10 L 100 R 100000 R 123 L 101 ? 123 L 10 R 115 ? 100 R 110 ? 115
output
0 2 1

Note

Let's take a look at the first example and let's consider queries:

1. The shelf will look like $[1]$;
2. The shelf will look like $[1, 2]$;
3. The shelf will look like $[1, 2, 3]$;

4. The shelf looks like $[1, \mathbf{2}, 3]$ so the answer is 1;
5. The shelf will look like $[4, 1, 2, 3]$;
6. The shelf looks like $[4, \mathbf{1}, 2, 3]$ so the answer is 1;
7. The shelf will look like $[5, 4, 1, 2, 3]$;
8. The shelf looks like $[5, 4, \mathbf{1}, 2, 3]$ so the answer is 2.

Let's take a look at the second example and let's consider queries:

1. The shelf will look like $[100]$;
2. The shelf will look like $[100, 100000]$;
3. The shelf will look like $[100, 100000, 123]$;
4. The shelf will look like $[101, 100, 100000, 123]$;
5. The shelf looks like $[101, 100, 100000, \mathbf{123}]$ so the answer is 0;
6. The shelf will look like $[10, 101, 100, 100000, 123]$;
7. The shelf will look like $[10, 101, 100, 100000, 123, 115]$;
8. The shelf looks like $[10, 101, \mathbf{100}, 100000, 123, 115]$ so the answer is 2;
9. The shelf will look like $[10, 101, 100, 100000, 123, 115, 110]$;
10. The shelf looks like $[10, 101, 100, 100000, 123, \mathbf{115}, 110]$ so the answer is 1.

F. Boxes Packing

time limit per test: 2.0 s
memory limit per test: 256 megabytes
input: standard input
output: standard output

Maksim has n objects and m boxes, each box has size exactly k. Objects are numbered from 1 to n in order from left to right, the size of the i-th object is a_i.

Maksim wants to pack his objects into the boxes and he will pack objects by the following algorithm: he takes one of the empty boxes he has, goes from left to right through the objects, and if the i-th object fits in the current box (the remaining size of the box is greater than or equal to a_i), he puts it in the box, and the remaining size of the box decreases by a_i. Otherwise he takes the new empty box and continues the process above. If he has no empty boxes and there is at least one object not in some box then Maksim cannot pack the chosen set of objects.

Maksim wants to know the maximum number of objects he can pack by the algorithm above. To reach this target, **he will throw out the leftmost object from the set until the remaining set of objects can be packed in boxes he has**. Your task is to say the maximum number of objects Maksim can pack in boxes he has.

Each time when Maksim tries to pack the objects into the boxes, he will make empty all the boxes he has before do it (and the relative order of the remaining set of objects will not change).

Input

The first line of the input contains three integers n, m, k ($1 \le n, m \le 2 \cdot 10^5$, $1 \le k \le 10^9$) — the number of objects, the number of boxes and the size of each box.

The second line of the input contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le k$), where a_i is the size of the i-th object.

Output

Print the maximum number of objects Maksim can pack using the algorithm described in the problem statement.

Examples

input
5 2 6 5 2 1 4 2
output
4

input
5 1 4 4 2 3 4 1
output
1

input
5 3 3 1 2 3 1 1
output
5

Note

In the first example Maksim can pack only 4 objects. Firstly, he tries to pack all the 5 objects. Distribution of objects will be $[5], [2, 1]$. Maxim cannot pack the next object in the second box and he has no more empty boxes at all. Next he will throw out the first object and the objects distribution will be $[2, 1], [4, 2]$. So the answer is 4.

In the second example it is obvious that Maksim cannot pack all the objects starting from first, second, third and fourth (in all these cases the distribution of objects is $[4]$), but he can pack the last object ($[1]$).

In the third example Maksim can pack all the objects he has. The distribution will be $[1, 2], [3], [1, 1]$.

G. Make It Connected

time limit per test: 3.0 s
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given an undirected graph consisting of n vertices. A number is written on each vertex; the number on vertex i is a_i. Initially there are no edges in the graph.

You may add some edges to this graph, but you have to pay for them. The cost of adding an edge between vertices x and y is $a_x + a_y$ coins. There are also m special offers, each of them is denoted by three numbers x, y and w, and means that you can add an edge connecting vertices x and y and pay w coins for it. You don't have to use special offers: if there is a pair of vertices x and y that has a special offer associated with it, you still may connect these two vertices paying $a_x + a_y$ coins for it.

What is the minimum number of coins you have to spend to make the graph connected? Recall that a graph is connected if it's possible to get from any vertex to any other vertex using only the edges belonging to this graph.

Input

The first line contains two integers n and m ($1 \le n \le 2 \cdot 10^5$, $0 \le m \le 2 \cdot 10^5$) — the number of vertices in the graph and the number of special offers, respectively.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^{12}$) — the numbers written on the vertices.

Then m lines follow, each containing three integers x, y and w ($1 \le x, y \le n$, $1 \le w \le 10^{12}$, $x \ne y$) denoting a special offer: you may add an edge connecting vertex x and vertex y, and this edge will cost w coins.

Output

Print one integer — the minimum number of coins you have to pay to make the graph connected.

Examples

input
3 2 1 3 3 2 3 5 2 1 1
output
5

input
4 0 1 3 3 7
output
16

input
5 4 1 2 3 4 5 1 2 8 1 3 10 1 4 7 1 5 15
output
18

Note

In the first example it is possible to connect 1 to 2 using special offer 2, and then 1 to 3 without using any offers.

In next two examples the optimal answer may be achieved without using special offers.

Codeforces (c) Copyright 2010-2022 Mike Mirzayanov
The only programming contests Web 2.0 platform