Codeforces Round #Pi (Div. 2)

A. Lineland Mail

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

All cities of Lineland are located on the Ox coordinate axis. Thus, each city is associated with its position x_i — a coordinate on the Ox axis. No two cities are located at a single point.

Lineland residents love to send letters to each other. A person may send a letter only if the recipient lives in another city (because if they live in the same city, then it is easier to drop in).

Strange but true, the cost of sending the letter is exactly equal to the distance between the sender's city and the recipient's city.

For each city calculate two values min_i and max_i, where min_i is the minimum cost of sending a letter from the i-th city to some other city, and max_i is the the maximum cost of sending a letter from the i-th city to some other city

Input

The first line of the input contains integer n ($2 \le n \le 10^5$) — the number of cities in Lineland. The second line contains the sequence of n distinct integers $x_1, x_2, ..., x_n$ ($-10^9 \le x_i \le 10^9$), where x_i is the x-coordinate of the i-th city. All the x_i's are distinct and follow in **ascending** order.

Output

Print n lines, the i-th line must contain two integers min_i, max_i, separated by a space, where min_i is the minimum cost of sending a letter from the i-th city, and max_i is the maximum cost of sending a letter from the i-th city.

Sample test(s)

input
4 -5 -2 2 7
output
3 12 3 9 4 7 5 12

input
2 -1 1
output
2 2 2 2

B. Berland National Library

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

Berland National Library has recently been built in the capital of Berland. In addition, in the library you can take any of the collected works of Berland leaders, the library has a *reading room*.

Today was the pilot launch of an automated reading room visitors' accounting system! The scanner of the system is installed at the entrance to the reading room. It records the events of the form "reader entered room", "reader left room". Every reader is assigned a *registration number* during the registration procedure at the library — it's a unique integer from 1 to 10^6. Thus, the system logs events of two forms:

- "+ r_i" — the reader with registration number r_i entered the room;
- "− r_i" — the reader with registration number r_i left the room.

The first launch of the system was a success, it functioned for some period of time, and, at the time of its launch and at the time of its shutdown, the reading room may already have visitors.

Significant funds of the budget of Berland have been spent on the design and installation of the system. Therefore, some of the citizens of the capital now demand to explain the need for this system and the benefits that its implementation will bring. Now, the developers of the system need to urgently come up with reasons for its existence.

Help the system developers to find the minimum possible capacity of the reading room (in visitors) using the log of the system available to you.

Input

The first line contains a positive integer n ($1 \le n \le 100$) — the number of records in the system log. Next follow n events from the system journal in the order in which the were made. Each event was written on a single line and looks as "+ r_i" or "− r_i", where r_i is an integer from 1 to 10^6, the registration number of the visitor (that is, distinct visitors always have distinct registration numbers).

It is guaranteed that the log is not contradictory, that is, for every visitor the types of any of his two consecutive events are distinct. Before starting the system, and after stopping the room may possibly contain visitors.

Output

Print a single integer — the minimum possible capacity of the reading room.

Sample test(s)

input
6 + 12001 - 12001 - 1 - 1200 + 1 + 7
output
3

input
2 - 1 - 2
output
2

input
2 + 1 - 1
output
1

Note

In the first sample test, the system log will ensure that at some point in the reading room were visitors with registration numbers 1, 1200 and 12001. More people were not in the room at the same time based on the log. Therefore, the answer to the test is 3.

C. Geometric Progression

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

Polycarp loves geometric progressions very much. Since he was only three years old, he loves only the progressions of length three. He also has a favorite integer k and a sequence a, consisting of n integers.

He wants to know how many subsequences of length three can be selected from a, so that they form a geometric progression with common ratio k.

A subsequence of length three is a combination of three such indexes i_1, i_2, i_3, that $1 \le i_1 < i_2 < i_3 \le n$. That is, a subsequence of length three are such groups of three elements that are not necessarily consecutive in the sequence, but their indexes are strictly increasing.

A geometric progression with common ratio k is a sequence of numbers of the form $b \cdot k^0, b \cdot k^1, ..., b \cdot k^{r-1}$.

Polycarp is only three years old, so he can not calculate this number himself. Help him to do it.

Input

The first line of the input contains two integers, n and k ($1 \le n, k \le 2 \cdot 10^5$), showing how many numbers Polycarp's sequence has and his favorite number.

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

Output

Output a single number — the number of ways to choose a subsequence of length three, such that it forms a geometric progression with a common ratio k.

Sample test(s)

input
5 2 1 1 2 2 4
output
4

input
3 1 1 1 1
output
1

input
10 3 1 2 6 2 3 6 9 18 3 9
output
6

Note

In the first sample test the answer is four, as any of the two 1s can be chosen as the first element, the second element can be any of the 2s, and the third element of the subsequence must be equal to 4.

D. One-Dimensional Battle Ships

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

Alice and Bob love playing one-dimensional battle ships. They play on the field in the form of a line consisting of n square cells (that is, on a $1 \times n$ table).

At the beginning of the game Alice puts k ships on the field without telling their positions to Bob. Each ship looks as a $1 \times a$ rectangle (that is, it occupies a sequence of a consecutive squares of the field). The ships cannot intersect and even touch each other.

After that Bob makes a sequence of "shots". He names cells of the field and Alice either says that the cell is empty ("miss"), or that the cell belongs to some ship ("hit").

But here's the problem! Alice like to cheat. May be that is why she responds to each Bob's move with a "miss".

Help Bob catch Alice cheating — find Bob's first move, such that after it you can be sure that Alice cheated.

Input

The first line of the input contains three integers: n, k and a ($1 \le n, k, a \le 2 \cdot 10^5$) — the size of the field, the number of the ships and the size of each ship. It is guaranteed that the n, k and a are such that you can put k ships of size a on the field, so that no two ships intersect or touch each other.

The second line contains integer m ($1 \le m \le n$) — the number of Bob's moves.

The third line contains m distinct integers $x_1, x_2, \ldots, x_m$, where x_i is the number of the cell where Bob made the i-th shot. The cells are numbered from left to right from 1 to n.

Output

Print a single integer — the number of such Bob's first move, after which you can be sure that Alice lied. Bob's moves are numbered from 1 to m in the order the were made. If the sought move doesn't exist, then print "-1".

Sample test(s)

input
11 3 3 5 4 8 6 1 11
output
3

input
5 1 3 2 1 5
output
-1

input
5 1 3 1 3
output
1

E. President and Roads

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

Berland has n cities, the capital is located in city s, and the historic home town of the President is in city t ($s \neq t$). The cities are connected by one-way roads, the travel time for each of the road is a positive integer.

Once a year the President visited his historic home town t, for which his motorcade passes along some path from s to t (he always returns on a personal plane). Since the president is a very busy man, he always chooses the path from s to t, along which he will travel the fastest.

The ministry of Roads and Railways wants to learn for each of the road: whether the President will definitely pass through it during his travels, and if not, whether it is possible to repair it so that it would definitely be included in the shortest path from the capital to the historic home town of the President. Obviously, the road can not be repaired so that the travel time on it was less than one. The ministry of Berland, like any other, is interested in maintaining the budget, so it wants to know the minimum cost of repairing the road. Also, it is very fond of accuracy, so it repairs the roads so that the travel time on them is always a positive integer.

Input

The first lines contain four integers n, m, s and t ($2 \leq n \leq 10^5$; $1 \leq m \leq 10^5$; $1 \leq s, t \leq n$) — the number of cities and roads in Berland, the numbers of the capital and of the Presidents' home town ($s \neq t$).

Next m lines contain the roads. Each road is given as a group of three integers a_i, b_i, l_i ($1 \leq a_i, b_i \leq n$; $a_i \neq b_i$; $1 \leq l_i \leq 10^6$) — the cities that are connected by the i-th road and the time needed to ride along it. The road is directed from city a_i to city b_i.

The cities are numbered from 1 to n. Each pair of cities can have multiple roads between them. It is guaranteed that there is a path from s to t along the roads.

Output

Print m lines. The i-th line should contain information about the i-th road (the roads are numbered in the order of appearance in the input).

If the president will definitely ride along it during his travels, the line must contain a single word "YES" (without the quotes).

Otherwise, if the i-th road can be repaired so that the travel time on it remains positive and then president will definitely ride along it, print space-separated word "CAN" (without the quotes), and the minimum cost of repairing.

If **we can't make the road be such that president will definitely ride along it**, print "NO" (without the quotes).

Sample test(s)

input

```
6 7 1 6
1 2 2
1 3 10
2 3 7
2 4 8
3 5 3
4 5 2
5 6 1
```

output

```
YES
CAN 2
CAN 1
CAN 1
CAN 1
CAN 1
YES
```

input

```
3 3 1 3
1 2 10
2 3 10
1 3 100
```

output

```
YES
YES
CAN 81
```

input

```
2 2 1 2
1 2 1
1 2 2
```

output

```
YES
NO
```

Note

The cost of repairing the road is the difference between the time needed to ride along it before and after the repairing.

In the first sample president initially may choose one of the two following ways for a ride: $1 \to 2 \to 4 \to 5 \to 6$ or $1 \to 2 \to 3 \to 5 \to 6$.

F. Mausoleum

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

King of Berland Berl IV has recently died. Hail Berl V! As a sign of the highest achievements of the deceased king the new king decided to build a mausoleum with Berl IV's body on the main square of the capital.

The mausoleum will be constructed from $2n$ blocks, each of them has the shape of a cuboid. Each block has the bottom base of a 1×1 meter square. Among the blocks, exactly two of them have the height of one meter, exactly two have the height of two meters, ..., exactly two have the height of n meters.

The blocks are arranged in a row without spacing one after the other. Of course, not every arrangement of blocks has the form of a mausoleum. In order to make the given arrangement in the form of the mausoleum, it is necessary that when you pass along the mausoleum, from one end to the other, the heights of the blocks first were *non-decreasing* (i.e., increasing or remained the same), and then — *non-increasing* (decrease or remained unchanged). It is possible that any of these two areas will be omitted. For example, the following sequences of block height meet this requirement:

- $[1, 2, 2, 3, 4, 4, 3, 1]$;
- $[1, 1]$;
- $[2, 2, 1, 1]$;
- $[1, 2, 3, 3, 2, 1]$.

Suddenly, k more requirements appeared. Each of the requirements has the form: "$h[x_i]$ $\texttt{sign}_i$ $h[y_i]$", where $h[t]$ is the height of the t-th block, and a $\texttt{sign}_i$ is one of the five possible signs: '=' (equals), '<' (less than), '>' (more than), '<=' (less than or equals), '>=' (more than or equals). Thus, each of the k additional requirements is given by a pair of indexes x_i, y_i ($1 \le x_i, y_i \le 2n$) and sign $\texttt{sign}_i$.

Find the number of possible ways to rearrange the blocks so that both the requirement about the shape of the mausoleum (see paragraph 3) and the k additional requirements were met.

Input

The first line of the input contains integers n and k ($1 \le n \le 35$, $0 \le k \le 100$) — the number of pairs of blocks and the number of additional requirements.

Next k lines contain listed additional requirements, one per line in the format "x_i $\texttt{sign}_i$ y_i" ($1 \le x_i, y_i \le 2n$), and the sign is on of the list of the five possible signs.

Output

Print the sought number of ways.

Sample test(s)

input
3 0
output
9

input
3 1 2 > 3
output
1

input
4 1 3 = 6
output
3