Problem A. Solving problems

Input file:	`standard input` или `input.txt`
Output file:	`standard output` или `output.txt`
Time limit:	1 second
Memory limit:	512 megabytes

Anton is in the 6th grade and actively preparing for programming competitions. At the moment, he has solved n problems, and he solved the i-th problem at hour t_i.

There are m time zones, numbered from 0 to $m-1$. In each time zone, one day consists of m consecutive hours. In the k-th time zone, the d-th day consists of hours numbered from $d \cdot m + k$ to $(d+1) \cdot m + k - 1$ (inclusive). Note that d can be negative.

At the beginning of the year, Anton set a goal to solve at least one problem every day. Now he wants to check if there is such a time zone and two days l and r such that on any of these days, he solved at least one problem, and any solved problem was solved exactly on one of the days from l to r in this time zone.

Help Anton find such a time zone or determine that it does not exist. If there are several suitable time zones, find the minimum one.

Input

The first line contains two integers, n and m ($1 \le n \le 200\,000$, $1 \le m \le 10^9$) — the number of problems solved and the number of hours in each day, respectively.

The second line contains n integers $t_1,\ t_2,\ t_3,\ \ldots,\ t_n$ ($0 \le t_i \le 10^9$, $t_i \le t_{i+1}$) — the submission time of each problem in non-decreasing order.

Output

Output one integer — the minimum number of the suitable time zone, or -1 if it does not exist.

Examples

standard input	standard output
3 3 4 5 10	2
4 5 2 4 14 17	-1
6 3 1 2 6 10 11 12	2

Note

In the first example, the day lasts for 3 hours, and Anton solved 3 problems: in hours 4, 5, and 10, respectively.

- In time zone 0, Anton's submissions fall on days 1, 1, and 3, respectively. Since Anton did not make any submissions on day 2, this time zone is not suitable.
- In time zone 1, Anton's submissions fall on days 1, 1, and 3, respectively. Since Anton did not make any submissions on day 2, this time zone is not suitable.
- In time zone 2, Anton's submissions fall on days 0, 1, and 2, respectively. The days form a continuous interval, so this time zone is suitable. Since it is the minimum among the suitable ones, 2 is the answer to the problem.

In the second example, none of the time zones is suitable.

In the third example:

- In time zone 0, Anton's submissions fall on days 0, 0, 2, 3, 3, and 4, respectively. Since Anton did not make any submissions on day 1, this time zone is not suitable.
- In time zone 1, Anton's submissions fall on days 0, 0, 1, 3, 3, and 3, respectively. Since Anton did not make any submissions on day 2, this time zone is not suitable.
- In time zone 2, Anton's submissions fall on days -1, 0, 1, 2, 3, and 3, respectively. The days form a continuous interval from -1 to 3, so this time zone is the answer.

Scoring

The tests for this problem consist of 5 groups. Points for each group are awarded only if all the tests in that group and some tests from the previous groups pass.

Group	Score	Additional constaints			Required groups	Comment
		n	m	t_i		
0	0	–	–	–	–	Samples.
1	18	$n \le 500$	$m \le 100$	–	0	
2	19	–	$m \le 100$	–	0, 1	
3	16	$n \le 500$	–	$t_i \le 500$	0	
4	21	$n \le 5000$	–	$t_i \le 500$	0, 3	
5	26	–	–	–	0 – 4	

Problem B. Yet another queries

Input file:	`standard input` или `input.txt`
Output file:	`standard output` или `output.txt`
Time limit:	1.5 seconds
Memory limit:	512 megabytes

You are given an array $a_0,\ a_1,\ \ldots,\ a_{2^n-1}$ consisting of 2^n elements. Note that the elements are numbered from zero.

You have to process q queries to this array of two types:

- $1\ l\ r\ k\ v$. In this case, the elements $a_{l\oplus k}$, $a_{(l+1)\oplus k}$, …, $a_{r\oplus k}$ are assigned value v.
- $2\ l\ r\ k$. In this case you need to calculate the sum $a_{l\oplus k} + a_{(l+1)\oplus k} + \cdots + a_{r\oplus k}$.

Here «$\oplus$» denotes the bitwise XOR operation.

Input

The first line contains a single integer n ($0 \le n \le 20$).

The second line contains 2^n integers $a_0,\ a_1,\ \ldots,\ a_{2^n-1}$ ($0 \le a_i \le 10^7$) – the elements of the array.

The third line contains a single integer q ($1 \le q \le 10^6$) – the number of queries.

The next q lines contain the query descriptions. The i-th line contains an integer t_i ($1 \le t_i \le 2$).

- If $t_i = 1$, then the line contains four integers l_i, r_i, k_i, and v_i ($0 \le l_i \le r_i < 2^n$, $0 \le k_i < 2^n$, $0 \le v_i \le 10^9$). In this case, we need to assign the value v_i to each of the elements $a_{l_i\oplus k_i}$, $a_{(l_i+1)\oplus k_i}$, …, $a_{r_i\oplus k_i}$.
- If $t_i = 2$, then the line contains three integers l_i, r_i, and k_i ($0 \le l_i \le r_i < 2^n$, $0 \le k_i < 2^n$). In this case, we need to calculate the sum $a_{l_i\oplus k_i} + a_{(l_i+1)\oplus k_i} + \ldots + a_{r_i\oplus k_i}$.

Output

For each query of the second type, print the desired sum.

Example

standard input	standard output
3 1 5 8 4 3 7 8 4 3 2 4 7 1 1 1 5 3 8 2 4 7 1	22 26

Note

In the first query, $a_{4\oplus1} + a_{5\oplus1} + a_{6\oplus1} + a_{7\oplus1} = a_5 + a_4 + a_7 + a_6 = 22$.

In the second query, we need to replace elements $a_{1\oplus3}, a_{2\oplus3}, a_{3\oplus3}, a_{4\oplus3}$ and $a_{5\oplus3}$ with 8. The array after these changes will be $8, 8, 8, 4, 3, 7, 8, 8$.

In the last query, $a_{4\oplus1} + a_{5\oplus1} + a_{6\oplus1} + a_{7\oplus1} = 26$.

Scoring

The tests for this problem consist of 8 groups. Points for each group are awarded only if all the tests in that group and some tests from the previous groups pass. **Offline-testing** means that the results of testing your solution on this group will only be available after the competition ends.

Group	Score	Additional constaints				Required groups	Comment
		n	q	a_i	v_i		
0	0	–	–	–	–	–	Samples.
1	19	$n \le 10$	$q \le 1000$	–	–	0	
2	17	$n \le 17$	$q \le 200\,000$	–	–	–	k_i is a power of two. If $t_i = 1$, then $l_i = r_i$
3	11	–	–	$a_i \le 1$	–	–	$t_i = 2$
4	13	–	–	$a_i \le 1$	$v_i \le 1$	3	
5	12	$n \le 17$	$q \le 200\,000$	–	–	–	$t_i = 2$
6	9	$n \le 17$	$q \le 200\,000$	–	–	2	If $t_i = 1$, then $l_i = r_i$
7	10	$n \le 17$	$q \le 200\,000$	–	–	0, 1, 2, 5, 6	
8	9	–	–	–	–	0 – 7	**Offline-testing.**

Problem C. Legs warm-up exercise

Input file:	`standard input` или `input.txt`
Output file:	`standard output` или `output.txt`
Time limit:	6 seconds
Memory limit:	512 megabytes

Once upon a time, a boy Kirill tried hard to solve a problem with the intriguing name «Hands warm-up exercise». Whether or not he managed to solve the problem, we do not know. However, during his contemplation, Kirill came up with a new problem for you to solve. The condition of the problem is as follows:

In the country X, there are n cities connected by m roads. The set of roads has two interesting properties. Firstly, all roads are one-way. Secondly, even if the roads were bidirectional, there would be no more than one simple path between any pair of cities. It is easy to notice that without the orientation, the graph with the given n cities as vertices and m roads as edges is a forest (meaning that in each connected component, the number of edges is one less than the number of vertices).

The function $cost(u, v)$ is defined as follows: if there exists a path from u to v, then it is equal to the number of roads on that path, and if there is no path, then it is equal to 0.

The *beauty* of the country map is defined as $\sum_{u=1}^{n} \sum_{v=1}^{n} cost(u, v)$, which is the sum of the values of *cost* for all pairs of cities.

The government of country X loves to calculate various statistics. This time, it was decided to calculate the *beauty* of the current country map, as well as process q changes to the country map, recalculating its *beauty* after each change.

There are 4 types of changes:

1. Add a road from city u to city v.
2. Remove the road between cities u and v.
3. Reverse the orientation of the road between cities u and v.
4. Orient all roads on the path from u to v in the direction from u to v. It is guaranteed that this is possible, i.e. there exists a path from city u to city v along the roads, ignoring their orientation.

It is guaranteed that after each change, the set of cities and roads continues to form a forest.

Note that in this problem, the queries modify the country map and depend on previous modifications.

Input

The first line contains an integer g ($0 \le g \le 10$) — the test group.

The second line contains three integers n, m and q ($2 \le n \le 400\,000$, $0 \le m < n$, $1 \le q \le 400\,000$) — the number of cities, roads and changes, respectively.

Each of the next m lines contains two integers u and v ($1 \le u, v \le n$) — the roads on the initial map oriented from u to v.

Each of the next q lines contains three integers t, u, and v ($1 \le t \le 4$, $1 \le u, v \le n$) — the type of query and the pair of vertices that define it.

Output

In the first line, print the initial *beauty* of the country map.

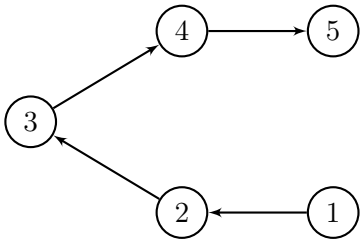
In the next q lines, print one number per line — the values of the *beauty* of the country map after each change.

Example

standard input	standard output
0 5 4 4 1 2 2 3 3 4 4 5 3 4 3 2 3 2 1 4 2 4 1 4	20 6 3 4 16

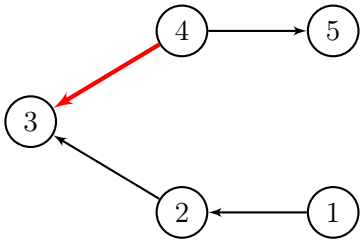
Note

In the test example, before any changes, the graph looks like this:

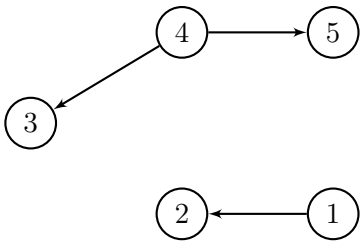


In this case, for example, $cost(1,2) = cost(2,3) = cost(3,4) = cost(4,5) = 1$, because there is a direct road between these pairs of cities. Similarly, $cost(1,3) = cost(2,4) = cost(3,5) = 2$, since the second city in each pair can only be reached in two roads from the first. Similarly, for example, $cost(2,1) = cost(4,1) = cost(5,3) = 0$, because the second city in the pair is not reachable from the first. Summing these values for all pairs of cities, we get 20.

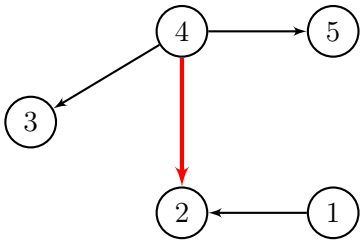
After that, we need to change the direction of the road between cities 3 and 4. Since it was previously from city 3 to city 4, now it should go from city 4 to city 3. After this, the answer to the problem is 6 and the graph looks like this:



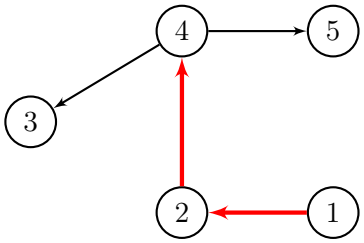
Next, the road between cities 2 and 3 is removed. After this, the answer to the problem is 3, and the graph looks like this:



After that, a road from city 4 to city 2 is added. After this, the answer to the problem is 4, and the graph looks like this:



After that, all roads between cities 1 and 4 are oriented from city 1 to city 4. After this, the answer to the problem is 16, and the graph looks like this:



Scoring

The tests for this problem consist of ten groups. Points for each group are awarded only if all the tests in that group and some tests from the previous groups pass. **Offline-testing** means that the results of testing your solution on this group will only be available after the competition ends.

Group	Score	Additional constaints		Required groups	Comment
		n, m, q	Changes types		
0	0	–	–	–	Samples.
1	10	$n, m, q \leq 100$	–	0	
2	8	$n, m, q \leq 5000$	–	0, 1	
3	11	$n, m, q \leq 100\,000$	1	–	
4	7	$n, m, q \leq 100\,000$	1, 2	3	
5	13	$n, m, q \leq 100\,000$	3	–	
6	9	$n, m, q \leq 100\,000$	3, 4	5	
7	12	$n, m, q \leq 100\,000$	–	–	$\lvert u - v \rvert = 1$ always, except for changes of type 4.
8	18	$n, m, q \leq 100\,000$	–	0 – 7	
9	6	$n, m, q \leq 200\,000$	–	0 – 8	**Offline-testing.**
10	6	–	–	0 – 9	**Offline-testing.**

Problem D. Hard problem

Input file:	`standard input` или `input.txt`
Output file:	`standard output` или `output.txt`
Time limit:	1 second
Memory limit:	512 megabytes

The boy Gena loves solving complex programming problems very much. Recently, he created a collection of n problems that he plans to solve. The problems have difficulty levels $a_1,\ a_2,\ \ldots,\ a_n$ and it is known that the difficulty levels of any two problems are different.

Gena is a very experienced programmer and can solve problems of any difficulty. He solves problems daily, following a specific strategy.

For each day, Gena has a fixed parameter m — the minimum difficulty level of the problems he is willing to solve. Every day, Gena goes through all the problems in his collection starting from the first to the last, and for each problem, he does the following action:

- If the current problem has already been solved, he skips it and moves on to the next one.
- If the difficulty level of the current problem is less than m, he skips it and moves on to the next one.
- If the previous conditions are not met and he has not solved any problems on the current day, he solves the current problem and moves on to the next one.
- If the previous conditions are not met and the last problem solved on the current day was easier than the current problem, he solves the current problem and moves on to the next one.
- If none of the previous conditions are met, Gena skips the problem and moves on to the next one.

In other words, Gena solves the problems in increasing order of difficulty every day, but only solves problems that he hasn't solved before and are of difficulty level greater than or equal to m.

On the first day, the minimum difficulty of the problems he is ready to solve is equal to t, and each subsequent day he will decrease m by the value of s. Thus, on the i-th day (starting from 1), Gena is ready to solve problems with difficulty level not less than $t - s \cdot (i - 1)$. Gena will repeat the algorithm described above daily until he solves all the problems.

The boy Lesha has been observing Gena's successes for a long time and wants to know the secret of his problem-solving strategy. Therefore, Lesha has q queries. For each query, it is required to check whether it is possible to rearrange the problems in Gena's selection so that he can solve the problem with difficulty level d_i as the p_i-th problem among all the problems solved on all days. Help Lesha answer all his queries.

Note that the queries are **independent**, meaning that different orders of problems can be used for different queries in Gena's selection.

Input

The first line contains three integers n, t and s ($1 \le n \le 200\,000$, $1 \le t, s \le 10^9$) — the number of problems in the selection, the minimum difficulty level on the first day, and the step of decreasing the difficulty level of the problems.

The second line contains n distinct integers $a_1,\ a_2,\ \ldots,\ a_n$ ($1 \le a_i \le 10^9$, $a_i < a_{i+1}$) — the difficulty levels of the problems.

The third line contains a single integer q ($1 \le q \le 200\,000$) — the number of queries.

Each of the next q lines contains two integers d_i and p_i ($1 \le p_i \le n$) — the parameters of the query: can the problems in the selection be rearranged so that the problem with **difficulty level** d_i is solved as the p_i-th problem. It is guaranteed that for each query, there exists a problem in the selection whose difficulty level is equal to d_i.

Output

For each query, print «Yes» (without quotes) if it is possible to rearrange the problems in the selection in a way that satisfies the query condition, and «No» (without quotes) otherwise.

Examples

standard input	standard output
5 10 2 4 5 9 10 12 5 10 2 10 3 10 4 5 5 12 2	Yes Yes No Yes Yes
7 4 2 2 3 5 6 9 10 11 4 5 6 11 7 2 2 10 7	Yes No Yes Yes

Note

Let's consider the first example and suppose that for the second query we can rearrange the problems in the selection in the following order: 12, 4, 5, 9, 10. Then Gena will solve the problems as follows:

1. On the first day, the minimum difficulty of the problems $m = 10$. The first problem in the selection satisfies this condition. Gena will solve it immediately. Among the remaining problems, there is a problem with difficulty 10, but since Gena has already solved a problem with difficulty 12 on the first day, he will skip the problem with difficulty 10. Thus, at the end of the first day, Gena will have solved only the problem with difficulty 12.
2. On the second day, the minimum difficulty of the problems $m = 8$. While going through the problems in the selection, Gena will skip problem 12 as he has already solved it. He will also skip problems 4 and 5, as their difficulty is less than m. Then he will encounter a problem with difficulty 9, and since he hasn't solved any problems on the second day yet, he will solve it. The next problem is of difficulty 10, and since its difficulty is greater than the difficulty of the last problem solved on that day, he will solve it. Thus, after the first two days, Gena will have solved the problems with difficulty 12, 9, 10 in that order.
3. On the third day, Gena will not solve any problems, as all the problems with difficulty at least $m = 6 = 10 - 2 \cdot 2$ have already been solved.
4. On the last fourth day, Gena will solve the remaining problems. In the end, he will have solved the problems with difficulty 12, 9, 10, 4, 5 in that order over all the days.

With the given rearrangement of tasks in the set, Gena will solve a task of difficulty 10 third in the order among all. Therefore, the answer to the second query is «Yes».

Scoring

The tests for this problem consist of 8 groups. Points for each group are awarded only if all the tests in that group and some tests from the previous groups pass.

Group	Score	Additional constaints				Required groups	Comment
		n	q	a_i	t		
0	0	–	–	–	–	–	Samples.
1	13	$n \leq 8$	–	$a_i \leq 10$	$t \leq 10$	–	
2	10	$n \leq 8$	–	–	–	0, 1	
3	14	$n \leq 500$	–	–	$t \leq 10$	$0, 1$	
4	19	$n \leq 500$	–	–	–	$0 - 3$	
5	9	–	$q = 1$	–	–	–	
6	1	–	–	–	$t = 1$	–	
7	9	–	–	–	–	–	$s = 10^9$
8	25	–	–	–	–	$0 - 7$	

Problem E. Colorful graph

Input file:	`standard input` или `input.txt`
Output file:	`standard output` или `output.txt`
Time limit:	1.5 seconds
Memory limit:	512 megabytes

You are given an undirected graph on n vertices with m edges numbered from 1 to m. Each edge is colored in one of k colors. The i-th edge connects the vertices v_i and u_i and has the color c_i.

Let's call a graph *good* if it is possible to leave exactly $n-1$ edge in it so that the graph is connected and there is at least one edge of each color among the edges left.

Given q changes in the colors of the edges of the graph. Each change is described by two numbers e_i and w_i and means that the color of the e_i-th edge becomes equal to w_i. After each change, determine if it is *good*.

Input

The first line contains three integers n, m and k ($2 \le n \le 100\,000$, $1 \le m \le 100\,000$, $1 \le k \le 8$) — the number of vertices, edges and colors respectively.

The following m lines contain a description of the edges. i-th of them contains three numbers v_i, u_i and c_i ($1 \le v_i, u_i \le n$, $1 \le c_i \le k$, $v_i \ne u_i$) — the endpoints of the i-th edge and its color, respectively.

The next line contains a single integer q ($1 \le q \le 100\,000$) — the number of changes.

The following q lines contain a description of the changes. i-th of them contains two integers e_i and w_i ($1 \le e_i \le m$, $1 \le w_i \le k$) — edge number and its new color.

It is guaranteed that there are no loops and multiple edges in the graph.

Output

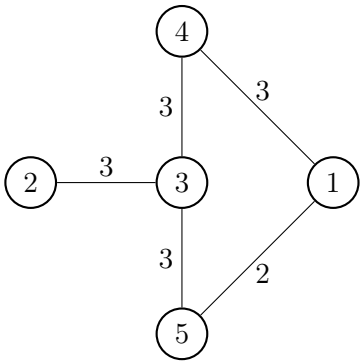
In the i-th line, print «`Yes`» (without quotes) if the graph after the i-th query is *good*, and «`No`» (without quotes) otherwise.

Examples

standard input	standard output
5 5 3 3 4 1 1 5 2 3 2 3 1 4 3 3 5 3 5 1 3 4 1 5 1 2 1 3 2	No Yes Yes No Yes
2 1 1 1 2 1 1 1 1	Yes

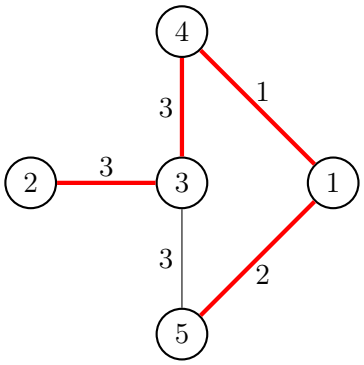
Note

In the first example, after the first change, the graph looks like this:



In this case, there are no edges of the color 1, so the condition of the problem cannot be met.

After the second change, the graph looks like this:



The edges that can be left are highlighted in red. This example is correct, since among these edges there are all the colors 1, 2 and 3, and they also form a connected graph.

Scoring

The tests for this problem consist of 8 groups. Points for each group are awarded only if all the tests in that group and some tests from the previous groups pass. **Offline-testing** means that the results of testing your solution on this group will only be available after the competition ends.

Group	Score	Additional constaints	Required groups	Comment
		k		
0	0	–	–	Samples.
1	10	$k \le 1$	–	
2	9	$k \le 2$	1	
3	15	$k \le 3$	0 – 2	
4	16	$k \le 4$	0 – 3	
5	14	$k \le 5$	0 – 4	
6	13	$k \le 6$	0 – 5	
7	12	$k \le 7$	0 – 6	
8	11	$k \le 8$	0 – 7	**Offline-testing.**

Problem F. Tournament

Input file:	`standard input` или `input.txt`
Output file:	`standard output` или `output.txt`
Time limit:	3 seconds
Memory limit:	512 megabytes

This is an interactive problem.

There is a tournament graph on n vertices numbered from 1 to n. A tournament graph is a directed graph where every edge connects two distinct vertices, and for any pair of distinct vertices u and v, there is exactly one edge oriented either from u to v or from v to u.

You only know the number of vertices in the graph, and in one query, you can find out the direction of the edge between any pair of vertices.

A vertex is called *good* if there is at most one outgoing edge from it. Your task is to find any *good* vertex or determine that there are no such vertices, using no more than 2000 queries.

Interaction Protocol

Your program will interact with the jury program using standard input and output streams. Each interaction will consist of solving the problem for multiple sets of input data.

First, your program should read two integers g and t ($0 \le g \le 10$, $1 \le t \le 100$) — the test group and the number of test cases within one interaction with the jury program. Then, t times, you need to perform the interaction to solve the problem for the input data set.

Let's consider the interaction protocol for one input data set.

First, your program should read one integer n ($1 \le n \le 500$) — the number of vertices in the hidden graph.

After that, you can make queries. For one query, output «`? u v`» ($1 \le u, v \le n$, $u \ne v$). In response to this, the jury program will print «`forward`» in a single line if the edge is directed from u to v, and «`backward`» if the edge is directed from v to u. For each test case, you can make at most 2000 queries.

To output the answer, print «`! u`» ($1 \le u \le n$ or $u = -1$), where u is a *good* vertex, or -1 if there is no such vertex. If the output answer is correct, the jury program will print «`OK`». After that, your program should immediately move on to process the next set of input data or terminate if it was the last graph. Otherwise, the jury program will print «`WRONG`». When reading this line, your program should immediately terminate its execution.

Please note that the constraints on n are given for each test case, there are no additional constraints on the sum of n for all sets of input data.

If you use «`cout << ... << endl`» in C++, «`System.out.println`» in Java, «`print`» in Python,«`writeln`» in Pascal, the output stream will be flushed automatically, so no additional action is required.

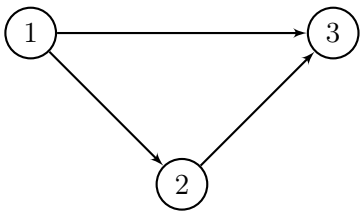
If you use a different method of output, it is recommended to flush the output stream. Please note that a new line should be output in any case. To flush the output stream, you can use «`fflush(stdout)`» in C++, «`flush(output)`» in Pascal, «`System.out.flush()`» in Java, «`sys.stdout.flush()`» in Python.

Example

standard input	standard output
0 2 3 forward backward OK 5 forward forward backward backward forward forward backward forward forward forward OK	 ? 1 2 ? 3 2 ! 3 ? 1 2 ? 1 3 ? 1 4 ? 1 5 ? 2 3 ? 2 4 ? 2 5 ? 3 4 ? 3 5 ? 4 5 ! -1

Note

In the first test case, since the jury program is not adaptive, the graph is predetermined and has the following form:



Therefore, in response to the query «? 1 2», the jury program outputted «forward», as the edge is directed from 1 to 2, and in response to the query «? 3 2», the jury program outputted «backward», as the edge is directed from 2 to 3. In this case, vertices 2 and 3 are *good*, while vertex 1 is not. Therefore, in response to the query «! 3», the jury program printed «OK».

In the second test case, there are multiple outgoing edges from each vertex, so in response to the query «! −1», the jury program printed «OK».

Scoring

The tests for this problem consist of 10 groups. Points for each group are awarded only if all the tests in that group and some tests from the previous groups pass. **Offline-testing** means that the results of testing your solution on this group will only be available after the competition ends.

In this problem, the jury program can act in two ways:

- Non-adaptive. That is, the structure of the graph is fixed in advance and does not adjust to the requests.
- Adaptive. That is, the structure of the graph can change during the interaction. However, it is guaranteed that there always exists at least one graph that fits the answers to all the already asked queries. In this case, the answer of your solution will be considered correct only if it is correct for all suitable graphs.

Group	Score	Additional constaints		Required groups	Comment
		n	Jury program		
0	0	–	Non-adaptive	–	Samples.
1	14	$n \le 63$	Adaptive	0	
2	8	–	Adaptive	–	Graph is *special*[1]
3	11	–	Adaptive	–	Graph is *special*[2]
4	12	–	Adaptive	–	Graph is *special*[3]
5	9	$n \le 100$	Adaptive	0, 1	
6	10	$n \le 400$	Non-adaptive	0	
7	9	$n \le 400$	Adaptive	0, 1, 5, 6	
8	8	$n \le 450$	Non-adaptive	0, 6	
9	10	–	Non-adaptive	0, 6, 8	
10	9	–	Adaptive	0 – 9	**Offline-testing.**

[1] A graph is called *special*[1] if there exists a permutation of vertices v_1, v_2, …, v_n such that for any $i < j$, the edge is oriented from vertex v_i to vertex v_j.

[2] A graph is called *special*[2] if for any vertex with a number $v > 2$, the edge is oriented from it to vertex 1 or to vertex 2, or to both vertices 1 and 2.

[3] A graph is called *special*[3] if $n \ge 4$ and there exists a permutation of vertices v_1, v_2, …, v_n such that the edge (v_i, v_{i+1}) is oriented from v_i to v_{i+1} and any other edge (v_i, v_j) $(i + 1 < j)$ is oriented from v_j to v_i.

Problem G. Space accident

Input file:	`standard input` или `input.txt`
Output file:	`standard output` или `output.txt`
Time limit:	1 second
Memory limit:	512 megabytes

You are a member of the crew of the spaceship «Hyperion» and recently you were involved in a fierce space battle from which you miraculously escaped. However, the reactor of your spaceship has suffered serious damage and overheated, causing the emergency cooling system to activate.

The reactor of the spaceship consists of n independent blocks, each of which has its own temperature expressed as an integer in degrees. In order for it to function stable, each block must have a **strictly** negative temperature. Thanks to scientific advances, the cooling system cools $n - 1$ blocks by B degrees in one cycle, and one remaining block by A degrees.

The cooling system was not damaged, but due to the spaceship's computer being damaged, it cannot calculate the minimum number of cycles required to fully cool down the reactor. Only you can solve this challenging task.

Input

The first line contains three integers n, A and B ($1 \le n \le 100\,000$, $1 \le A, B \le 10^9$) — the number of blocks in the reactor and the parameters A and B.

The next line contains n integers $t_1,\ t_2,\ \ldots,\ t_n$ ($-10^9 \le t_i \le 10^9$), where t_i — the temperature of the i-th unit in the reactor.

Output

Print a single number — the minimum number of cycles until the reactor is completely cooled.

Examples

standard input	standard output
5 1 2 1 2 3 4 5	3
3 42 42 -273 -273 -273	0
1 3 2 9	4
4 4 2 5 5 1 0	2

Note

In the fourth test case, the cooling system can cool the first block by 4 degrees on the first cycle, and the remaining blocks by 2 degrees each. As a result, the temperatures of the blocks will be as follows: $\{1, 3, -1, -2\}$. On the next cycle, the second block can be cooled by 4 degrees, after which the temperatures of all the blocks will become negative.

Scoring

The tests for this problem consist of 6 groups. Points for each group are awarded only if all the tests in that group and some tests from the previous groups pass.

Group	Score	Additional constaints				Required groups	Comment
		n	t_i	A	B		
0	0	–	–	–	–	–	Samples.
1	15	$n \leq 5$	$\|t_i\| \leq 5$	$A \leq 5$	$B \leq 5$	–	
2	16	–	–	$A = 1$	$B = 2$	–	
3	10	–	–	–	–	–	A=B
4	15	n = 2	–	–	–	–	
5	24	$n \leq 500$	$\|t_i\| \leq 500$	$A \leq 500$	$B \leq 500$	0, 1	
6	20	–	–	–	–	0 – 5	

Problem H. Lunch

Input file:	`standard input` или `input.txt`
Output file:	`standard output` или `output.txt`
Time limit:	2 seconds
Memory limit:	512 megabytes

You are given a convex polygon with N points. **It is not guaranteed** that no three points lie on the same line. There are M different *special* points inside the polygon. Additionally, a point C is given with coordinates (x_0, y_0), which lies inside the convex polygon but not on its boundary.

Alice and Bob play a game, taking turns starting with Alice. On each turn, a player must choose a vertex of the polygon different from C and move it to point C. If there is already a vertex at point C in the polygon, these vertices are merged. A move can only be made if there is a *special* point that lies inside the polygon before the move, but not in it afterwards. It is guaranteed that the point can never be on the boundary of the polygon after any number of moves.

After a move, the polygon is not required to be convex and can also become degenerate (i.e., become a segment). The game ends when a player cannot make a move.

There are also q changes of two types:

- $+$ x y, which means that the point with coordinates (x, y) becomes *special*. It is guaranteed that this point was not *special* before.
- $-$ x y, which means that the point with coordinates (x, y) stops being *special*. It is guaranteed that this point was *special* before.

After each modification, as well as at the beginning, determine which player should win if both of them play optimally. After each modification, the game starts with the initial polygon, taking into account the applied modifications to the special points.

Input

The first line contains three integers n, m and q ($3 \le n \le 10\,000$, $0 \le m \le 100\,000$, $0 \le q \le 1\,000\,000$) — the number of points in the polygon, the number of *special* points and the number of changes.

The second line contains two integers x_0 and y_0 ($-10^9 \le x_0, y_0 \le 10^9$) — coordinates of the point C. It is guaranteed that the point lies inside the given polygon and does not lie on its border.

Each of the next n lines contains two integers x_i and y_i ($-10^9 \le x_i, y_i \le 10^9$) — coordinates of the i-th point of the polygon. The points are given in a counterclockwise order.

Each of the next m lines contains two integers x_i and y_i ($-10^9 \le x_i, y_i \le 10^9$) — coordinates of the i-th *special* point. It is guaranteed that at any time the points are distinct and lie inside the polygon.

The following q lines contain a description of the requests. The i-th of them contains the symbol c and two integers x and y ($c =$ «+» or «-» (without quotes), $-10^9 \le x, y \le 10^9$) — description of the next request.

- If $c =$ «+», then the point (x, y) becomes *special*. It is guaranteed that this point was not *special* before.
- If $c =$ «-», then the point (x, y) ceases to be *special*. It is guaranteed that before that this point was *special*.

It is guaranteed that at any time after any number of correct moves, none of the singular points can be on the boundary of the polygon.

Output

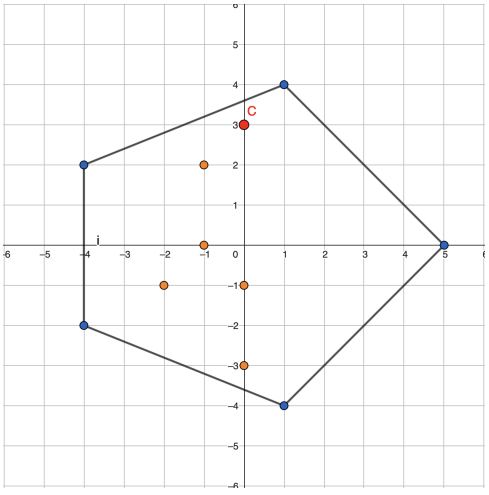
Print $q + 1$ lines. In the first line, print «`Alice`» (without quotes) if Alice wins before any modifications with optimal play, and «`Bob`» (without quotes) otherwise. Then, in the i-th line, print the winner of the game after the $(i - 1)$-th modification in a similar format.

Example

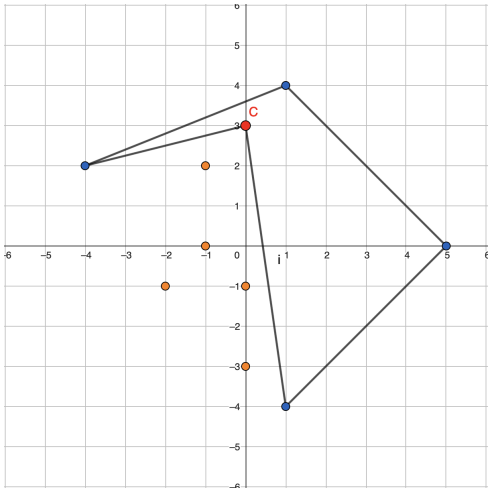
standard input	standard output
5 5 3 0 3 -4 -2 1 -4 5 0 1 4 -4 2 0 -1 -2 -1 -1 0 0 -3 -1 2 + 4 0 + -2 2 + 0 2	Alice Bob Bob Bob

Note

Consider the polygon before any modifications:



On the first move, Alice can move a vertex of the polygon with coordinates (-4, -2), then the polygon will look like this:



In this state of the polygon, Bob cannot make a move anymore, and therefore loses.

Scoring

The tests for this problem consist of 6 groups. Points for each group are awarded only if all the tests in that group and some tests from the previous groups pass. **Offline-testing** means that the results of testing your solution on this group will only be available after the competition ends.

Group	Score	Additional constaints			Required groups	Comment
		n	m	q		
0	0	–	–	–	–	Samples.
1	16	$n = 3$	$m \le 3$	$q = 0$	–	
2	13	$n \le 18$	$m \le 18$	$q \le 1$	1	
3	15	$n \le 18$	$m \le 18$	–	0 – 2	
4	17	$n \le 5000$	$m \le 5000$	$q \le 1$	1, 2	
5	21	$n \le 5000$	$m \le 100\,000$	$q \le 5000$	0 – 2, 4	
6	18	–	–	–	0 – 5	**Offline-testing.**

Problem I. Paired roads

Input file:	`standard input` или `input.txt`
Output file:	`standard output` или `output.txt`
Time limit:	4 seconds
Memory limit:	512 megabytes

In a certain country, there are n cities, between which no roads have been built yet. The i-th city has a population of w_i people. There are also $n-1$ roads that can be built. The i-th road, when built, will connect cities u_i and v_i, and its construction cost is s_i.

It is known that if all the roads are built, then from every city it will be possible to reach any other city using only these roads. In other words, the roads form a tree.

During each of the following k days, the following will happen: on the i-th day, a city c_i is chosen, as well as two distinct roads that have not been built yet, connecting this city to some other cities. After that, these two roads are constructed, and the price paid is equal to the sum of the construction costs of these two roads. The city c_i is then considered *central* on day i.

After k days, each city that is considered *central* at least once will generate profit equal to the number of people living in it.

We define *benefit* as the difference between the profit brought by the cities and the total cost of the constructed roads. Find the maximum possible *benefit*.

Input

The first line contains three integers n, k, and t ($3 \le n \le 200\,000$, $1 \le k \le \frac{n-1}{2}$, $0 \le t \le 1$) — the number of cities, the number of pairs of roads to be built, and integer t, which is equal to 1 if it's required to find which roads to build, and 0 otherwise.

The second line contains n integers w_1, w_2, …, w_n ($1 \le w_i \le 10^8$) — population of each city.

Each of the next $n-1$ lines contains three integers u_i, v_i, and s_i ($1 \le u_i, v_i \le n$, $1 \le s_i \le 10^8$) — the cities connected by the i-th road and the cost of building it.

It is guaranteed that the roads form a tree, and that it is possible to build k pairs of roads that satisfy the given condition.

Output

In the first line, print a single integer — the maximum *benefit* that can be obtained after building exactly k pairs of roads.

If $t = 1$, then in the next k lines, print k triples of numbers c_i, x_i, and y_i ($1 \le c_i, x_i, y_i \le n$) — the description of the pairs of roads that need to be built. Such a triple represents a pair of roads (c_i, x_i) and (c_i, y_i).

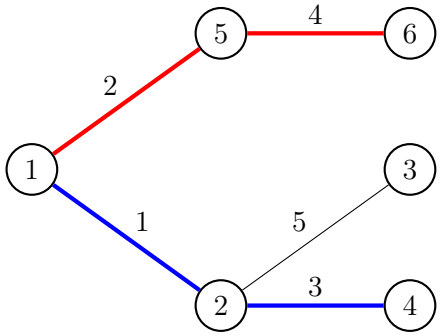
If there are multiple ways to achieve the maximum *benefit*, output any of them.

Examples

standard input	standard output
6 2 1 1 2 3 4 5 6 1 2 1 2 3 5 2 4 3 1 5 2 5 6 4	-3 5 6 1 2 4 1
8 3 0 4 5 1 2 3 1 3 5 2 1 15 7 1 5 4 8 1 8 5 2 7 8 1 6 7 5 3 7 7	-13

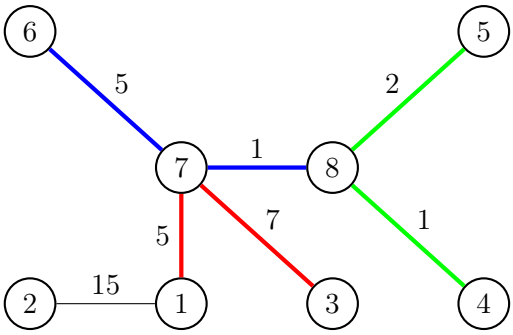
Note

In the first test from the problem statement, it is optimal to construct the following roads (roads from the same pair are marked with the same color):



The total cost of constructing these roads is equal to $2 + 4 + 1 + 3 = 10$. After construction, cities 2 and 5 will generate profits of 2 and 5 respectively. Thus, the *benefit* is $2 + 5 - 10 = -3$. It can be shown that it is not possible to obtain a better answer.

In the second test from the problem statement, it is optimal to construct the following roads:



The total cost of constructing these roads is equal to $5+1+2+1+5+7 = 21$. After construction, cities 7 and 8 will generate profits of 3 and 5 respectively. Thus, the *benefit* is $3+5-21 = -13$. It can be shown that it is not possible to obtain a better answer.

Scoring

The tests for this problem consist of 7 groups. Points for each group are awarded only if all the tests in that group and some tests from the previous groups pass. **Offline-testing** means that the results of testing your solution on this group will only be available after the competition ends.

Group	Score	Additional constaints		Required groups	Comment
		n	t		
0	0	–	–	–	Samples.
1	13	$n \le 200$	$t = 0$	–	
2	17	$n \le 2000$	$t = 0$	1	
3	12	$n \le 2000$	–	0 – 2	
4	19	–	$t = 0$	–	$u_i = i$, $v_i = i + 1$
5	11	–	–	4	$u_i = i$, $v_i = i + 1$
6	15	–	$t = 0$	1, 2, 4	**Offline-testing.**
7	13	–	–	0 – 6	**Offline-testing.**