Tsinghua Bootcamp 2024. Day 3

A. Another Problem about Queries on a Tree

1 second, 256 megabytes

Once, Mike was walking in his home town and discovered a remarkable *tree* with n vertices, numbered with integers from 1 to n. The tree is rooted to the vertex numbered 1. Recall that a tree is a connected undirected graph without cycles.

In each vertex of the tree v, two integers t_v and d_v are recorded. Initially, there are pieces in some vertices of the tree.

Mike gave you q queries, each characterized by an integer c. To execute a query, you must consider all vertices that contain a piece and for which $t_v = c$. Let these vertices be numbered $v_1, v_2, \ldots, v_k$. After that, for each vertex **simultaneously** and independently, the process described below is executed.

Consider vertex v_i and the path from this vertex to the root of the tree (that is, to the vertex numbered 1). Let this path consist of vertices $u_0, u_1, \ldots, u_m$, where $u_0 = v_i$, $u_m = 1$, and vertex u_i is an ancestor of vertex u_{i-1}. Remove the piece from vertex v_i and move it to the vertex numbered $u_{d_{v_i}}$. If $d_{v_i} > m$, the piece is moved to the root of the tree. For better understanding of the process, please refer to the examples below.

You must execute the queries sequentially and find the minimum query number after which all pieces will be at the root of the tree, or determine that after executing all of Mike's queries, not all pieces will be at the root of the tree.

Input

The first line contains two integers n and q ($2 \le n \le 200\,000$, $1 \le q \le 200\,000$) — the number of vertices in the tree and Mike's queries, respectively.

The second line contains $n - 1$ integers $p_2, p_3, \ldots, p_n$ ($1 \le p_i < i$). These numbers indicate that the ancestor of vertex i in the tree is vertex p_i.

The third line contains n integers t_i ($1 \le t_i \le n$).

The fourth line contains n integers d_i ($1 \le d_i \le n$).

The fifth line contains n integers a_i ($a_i \in \{0, 1\}$). If $a_i = 1$, then there is initially a piece in the i-th vertex of the tree. It is guaranteed that there is at least one piece in the tree that is not at the root.

The sixth line contains q integers c_i ($1 \le c_i \le n$) — the parameters of the queries.

Output

If there exists a minimum query number i ($1 \le i \le q$) after which all pieces will be at the root of the tree, output the number i.

Otherwise, output the number -1.

input
5 6 1 1 1 4 1 5 3 5 3 5 3 1 3 2 1 1 0 1 0 3 4 1 5 4 2
output
4

input
5 1 1 1 1 4 1 5 3 5 3 5 3 1 3 2 1 1 0 1 0 2
output
-1

B. Partitioning into Triples

1 second, 512 megabytes

For her birthday, Mary was gifted an array a of n natural numbers, where each number is in the range from 1 to m inclusive. Mary loves the number three, so the length of the array is divisible by three.

Mary decided to group the numbers into *triples*: each triple must consist of either three identical numbers or three consecutive numbers. In other words, each triple has either the form (x, x, x) or $(x, x + 1, x + 2)$, where x is some natural number.

Mary wants to play with the gifted array, and she is interested in the number of ways to partition the numbers of this array into such triples. Two partitioning methods are considered different if there is no one-to-one correspondence between the triples of the first partitioning and the triples of the second partitioning such that the numbers within the corresponding triples are equal. Since the number of partitions can be large, Mary only needs to know the remainder modulo $10^9 + 7$.

Help Mary count the number of ways to partition the numbers of the gifted array into triples modulo $10^9 + 7$.

Input

The first line contains two integers n and m ($1 \le n \le 5000$, $1 \le m \le 5000$, $n = 3 \cdot k$ for some natural k).

The second line contains n integers a_i — the numbers of the array ($1 \le a_i \le m$).

Output

In a single line, output one number — the number of ways to partition the numbers of the array into triples modulo $10^9 + 7$.

input
9 4 3 4 2 4 4 2 3 3 2
output
2

input
6 3 1 2 3 1 2 1
output
0

In the first example, the numbers can be partitioned into triples in two ways: $\{(2, 2, 2), (3, 3, 3), (4, 4, 4)\}$ and $\{(2, 3, 4), (2, 3, 4), (2, 3, 4)\}$.

C. Klyukalo

1 second, 256 megabytes

Klyukalo consists of N parts, each with its own standard — the i-th part should weigh s_i grams. If there is a klyukalo where the i-th part weighs a_i grams, its *deviation* can be calculated using the formula $\frac{|a_i - s_i|}{s_i}$. The *deviation* of the entire structure is calculated using the formula $\sum \frac{|a_i - s_i|}{s_i}$, which is the sum of the *deviations* of each part. The allowable deviation of the klyukalo according to the standard is K.

You are given a klyukalo. In one minute, you can either increase the weight of one part by 1 gram or decrease the weight of one part by 1 gram. What is the minimum time required to bring the given klyukalo to the standard with a deviation of no more than K?

Input
The first line contains two integers N and K — the number of parts in the klyukalo and the allowable deviation $(1 \le N \le 10^5, 0 \le K \le 10^9)$.

The second line contains N integers s_i — the standard weights of the parts $(1 \le s_i \le 10)$.

The third line contains N integers a_i — the weights of the parts in the given klyukalo $(1 \le a_i \le 10^9)$.

Output
Output the minimum number of minutes required to bring the given klyukalo to the standard with a deviation of no more than K.

input
3 1 1 2 1 2 4 3
output
3

In the example, you can reduce the weight of the first and third parts to the standard.

D. Another Problem about the Game with Stones

1 second, 256 megabytes

As a result of an unfortunate turn of events, Mike found himself on a deserted island. The first thing he did, of course, was to arrange stones to form a large S.O.S. sign on the beach. However, before help arrives, he needs something to entertain himself, so he decided to play with the remaining stones.

Mike arranged all the stones he had left into n piles such that the i-th pile contained exactly r_i stones. After that, the boy decided to take some number of stones from each pile, ensuring that the following conditions were met:

- Mike must take at least l_i and at most r_i stones from the i-th pile;
- The total number of taken stones must equal s.

After Mike completed this task, he wondered in how many ways he could take stones from the piles such that the described conditions were satisfied. Specifically, for each pile i, he wants to calculate how many ways he can choose a certain number of stones from the i-th pile so that it is possible to take a certain number of stones from the remaining piles while fulfilling the stated conditions.

Unfortunately or fortunately, help arrived too quickly, and Mike did not have time to find the answer to his question. Therefore, it is up to you to do this.

Input
The first line contains two integers n and s $(1 \le n \le 100\,000, 0 \le s \le 10^{18})$ — the number of piles of stones and the total number of stones taken from the piles.

Each of the following n lines contains two integers l_i and r_i ($0 \le l_i \le r_i \le 10^9$) — the minimum and maximum number of stones that Mike can take from the i-th pile.

Output
Output n integers $c_1, c_2, \ldots, c_n$, denoting the number of ways to choose a certain number of stones from the i-th pile so that it is possible to take a certain number of stones from the remaining piles to satisfy the conditions.

input
5 10 1 3 2 3 3 3 0 10 1 1
output
3 2 1 4 1

Consider the example from the statement.

From the first pile, you can choose 1, 2, or 3 stones. For each of these ways, you can choose a certain number of stones from the remaining piles such that a total of 10 stones is selected. For example, this can be done in the following ways: $1 + 2 + 3 + 3 + 1 = 10$, $2 + 2 + 3 + 2 + 1 = 10$, $3 + 2 + 3 + 1 + 1 = 10$.

From the second pile, you can choose 2 or 3 stones, for example, in the following ways: $1 + 2 + 3 + 3 + 1 = 10$, $1 + 3 + 3 + 2 + 1 = 10$.

From the third pile, you can choose 3 stones. There are no other options since $l_3 = r_3 = 3$.

From the fourth pile, you can choose 0, 1, 2, or 3 stones. The ways to choose 1, 2, or 3 stones have already been described above. The way to choose 0 stones looks like this: $3 + 3 + 3 + 0 + 1 = 10$.

From the fifth pile, you can choose 1 stone. There are no other options since $l_5 = r_5 = 1$.

E. Lada Malina

4 seconds, 256 megabytes

After long-term research and lots of experiments leading Megapolian automobile manufacturer "AutoVoz" released a brand new car model named "Lada Malina". One of the most impressive features of "Lada Malina" is its highly efficient environment-friendly engines.

Consider car as a point in Oxy plane. Car is equipped with k engines numbered from 1 to k. Each engine is defined by its velocity vector whose coordinates are (vx_i, vy_i) measured in distance units per day. An engine may be turned on at any level w_i, that is a real number between -1 and $+1$ (inclusive) that result in a term of $(w_i \cdot vx_i, w_i \cdot vy_i)$ in the final car velocity. Namely, the final car velocity is equal to

$$(w_1 \cdot vx_1 + w_2 \cdot vx_2 + ... + w_k \cdot vx_k,\quad w_1 \cdot vy_1 + w_2 \cdot vy_2 + ... + w_k \cdot vy_k)$$

Formally, if car moves with constant values of w_i during the whole day then its x-coordinate will change by the first component of an expression above, and its y-coordinate will change by the second component of an expression above. For example, if all w_i are equal to zero, the car won't move, and if all w_i are equal to zero except $w_1 = 1$, then car will move with the velocity of the first engine.

There are n factories in Megapolia, i-th of them is located in (fx_i, fy_i). On the i-th factory there are a_i cars "Lada Malina" that are ready for operation.

As an attempt to increase sales of a new car, "AutoVoz" is going to hold an international exposition of cars. There are q options of exposition location and time, in the i-th of them exposition will happen in a point with coordinates (px_i, py_i) in t_i days.

Of course, at the "AutoVoz" is going to bring as much new cars from factories as possible to the place of exposition. Cars are going to be moved by enabling their engines on some certain levels, such that at the beginning of an exposition car gets exactly to the exposition location.

However, for some of the options it may be impossible to bring cars from some of the factories to the exposition location by the moment of an exposition. Your task is to determine for each of the options of exposition location and time how many cars will be able to get there by the beginning of an exposition.

Input
The first line of input contains three integers k, n, q $(2 \le k \le 10$, $1 \le n \le 10^5$, $1 \le q \le 10^5)$, the number of engines of "Lada Malina", number of factories producing "Lada Malina" and number of options of an exposition time and location respectively.

The following k lines contain the descriptions of "Lada Malina" engines. The i-th of them contains two integers vx_i, vy_i ($-1000 \le vx_i, vy_i \le 1000$) defining the velocity vector of the i-th engine. Velocity vector can't be zero, i.e. at least one of vx_i and vy_i is not equal to zero. It is guaranteed that no two velosity vectors are collinear (parallel).

Next n lines contain the descriptions of factories. The i-th of them contains two integers fx_i, fy_i, a_i ($-10^9 \le fx_i, fy_i \le 10^9$, $1 \le a_i \le 10^9$) defining the coordinates of the i-th factory location and the number of cars that are located there.

The following q lines contain the descriptions of the car exposition. The i-th of them contains three integers px_i, py_i, t_i ($-10^9 \le px_i, py_i \le 10^9$, $1 \le t_i \le 10^5$) defining the coordinates of the exposition location and the number of days till the exposition start in the i-th option.

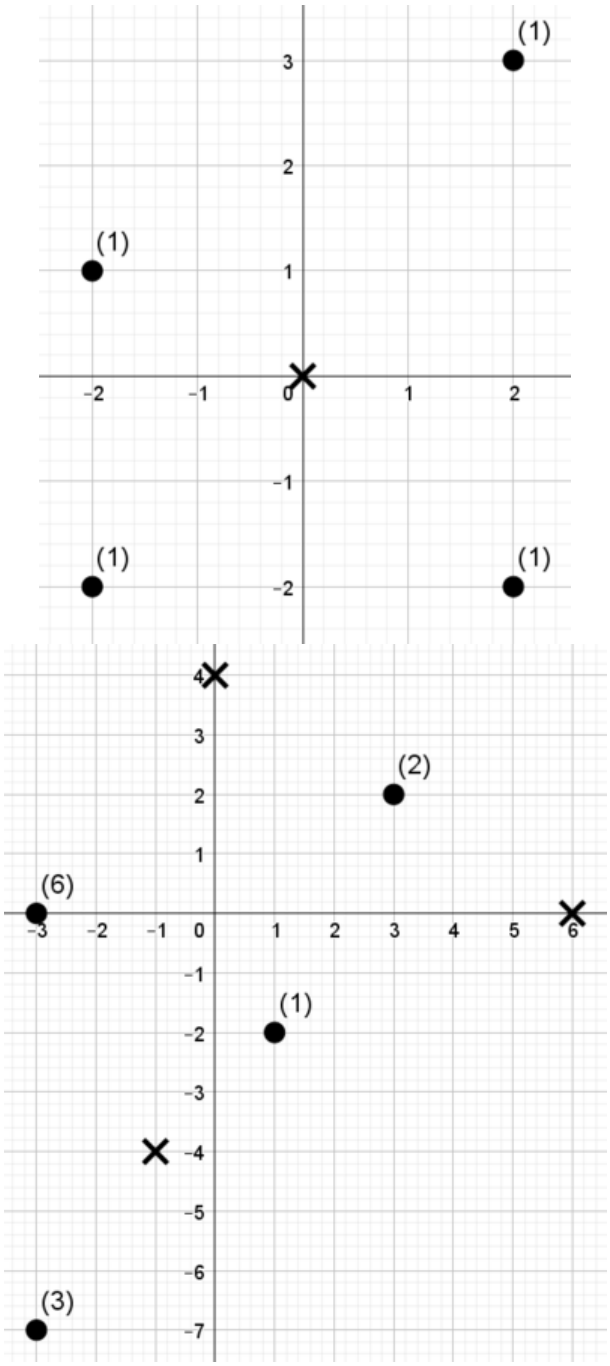
Output

For each possible option of the exposition output the number of cars that will be able to get to the exposition location by the moment of its beginning.

input
2 4 1 1 1 -1 1 2 3 1 2 -2 1 -2 1 1 -2 -2 1 0 0 2
output
3

input
3 4 3 2 0 -1 1 -1 -2 -3 0 6 1 -2 1 -3 -7 3 3 2 2 -1 -4 1 0 4 2 6 0 1
output
4 9 0

Images describing sample tests are given below. Exposition options are denoted with crosses, factories are denoted with points. Each factory is labeled with a number of cars that it has.



First sample test explanation:

- Car from the first factory is not able to get to the exposition location in time.
- Car from the second factory can get to the exposition in time if we set $w_1 = 0$, $w_2 = 1$.
- Car from the third factory can get to the exposition in time if we set $w_1 = \frac{1}{4}, w_2 = \frac{3}{4}$.
- Car from the fourth factory can get to the exposition in time if we set $w_1 = 1$, $w_2 = 0$.

F. Choosing a Capital

2 seconds, 512 megabytes

Given an undirected tree — a connected graph with n vertices and no cycles, and a number k. Fix a certain vertex s of the tree and call it the capital.

Orient the edges of the tree in the direction away from the capital. In other words, orient the edge (u, v) in the direction $u \to v$ if, when hanging the tree from vertex s, vertex u is the parent of vertex v. Note that with this orientation of the edges, every vertex is reachable from the capital.

Define the distance to vertex v in the graph as the minimum number of edges on the path from s to v. We call the *accessibility* of vertex s the maximum of the distances to all vertices.

You are allowed to add no more than k additional directed edges to the tree.

For each vertex s of the tree, determine what minimum *accessibility* can be achieved if vertex s is chosen as the capital.

Note that in some tests, the answer is required only for the first vertex.

Input

The first line contains three integers n, k, and t ($2 \le n \le 2 \cdot 10^5$, $1 \le k \le n - 1$, $n \cdot k \le 2 \cdot 10^5$, $0 \le t \le 1$) — the number of vertices in the tree, the limit on the maximum number of added edges, and the number t, which is equal to 0 if the answer is required only for vertex number 1, and equal to 1 otherwise.

Each of the following $n - 1$ lines contains two integers u_i, v_i ($1 \le u_i, v_i \le n$) — the edges of the tree.

It is guaranteed that the given edges form a tree.

Output

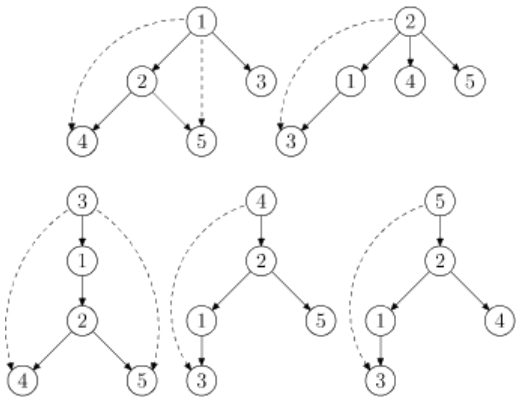
If $t = 0$, output a single integer: the minimum *accessibility* that can be achieved by choosing vertex number 1 as the capital and adding no more than k additional directed edges.

If $t = 1$, output n numbers: the i-th number equals the minimum *accessibility* that can be achieved by choosing vertex i as the capital and adding no more than k additional directed edges.

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

input
3 1 0 1 2 2 3
output
1

The figure illustrates the first example. The dashed lines indicate the added edges. For vertices 1 and 2, the minimum *accessibility* is 1, while for vertices 3, 4, and 5, the minimum *accessibility* is 2.



G. House Search

3 seconds, 1024 megabytes

This is an interactive run-twice problem. Your solution will be executed two times for each test.

In the old city of Lund, there is a street with n houses lined up, numbered with integers from 0 to $n - 1$. Emma lives in one of these houses, and her friends Anna and Bertil want to find out which house Emma lives in. Instead of simply telling her friends where she lives, Emma decided to play a game with them. Before the game starts, Anna and Bertil only know the number of houses on the street. At this point, Anna and Bertil can choose a positive integer k and agree on a strategy. After that, any communication is prohibited.

The game itself consists of two phases. In the first phase, Emma chooses the order of visiting the houses such that her house is the last one. Each house will be visited exactly once. Then she takes Anna to the houses in turn according to this order, without informing Anna of the order in advance. In each house that is not Emma's, Anna is allowed to write one integer from 1 to k on the front door of the house with chalk. In the last house they visit (Emma's house), Emma herself writes an integer from 1 to k on the door.

In the second phase of the game, Bertil walks down the street from house 0 to house $n - 1$ and reads all the numbers written on the doors by Anna and Emma. Now he wants to guess which house Emma lives in. He has two chances to guess the correct answer, and if he succeeds, he and Anna win the game. Otherwise, Emma wins the game.

Can you devise a strategy that guarantees Anna and Bertil will win the game?

Interaction

Your solution will be run twice for each test. In the first run, your solution will play the role of Anna, and in the second run, it will play the role of Bertil. The jury program, with which you will interact, will play the role of Emma.

At the beginning of each run, your program must read two integers p and n ($1 \le p \le 2$) — the run number and the number of houses on the street. In all tests, except for the example from the statement, $n = 100\,000$.

Then you must interact with the jury program, following the protocol below.

First run.

Your program must output one integer k ($1 \le k \le 7$) — the number chosen by Anna and Bertil.

After that, your program must read one integer i ($0 \le i < n$) $n - 1$ times — the number of the next visited house, and output one integer a_i ($1 \le a_i \le k$) — the number that Anna will write on the door of the i-th house.

It is guaranteed that each house number i, except for Emma's house number, will appear in the input exactly once. The order of visiting the houses is determined by the jury program and is not known to you in advance.

Do not forget to perform the `flush` operation after outputting the number k, as well as after outputting each value a_i.

Second run.

Your program must read n integers $a_0, a_1, \ldots, a_{n-1}$ ($1 \le a_i \le k$) — the numbers written on the 0-th, 1-st, ..., $(n - 1)$-th houses.

After that, you must guess the number of the house where Emma lives. To do this, you can output two integers s_1 and s_2 ($0 \le s_1, s_2 < n$) — the two guesses you made.

If Emma lives in house number s_1 or in house number s_2, the test will be considered successfully passed. Otherwise, the score for the test will be 0 points.

Do not forget to perform the `flush` operation after outputting the numbers s_1 and s_2.

The jury program, playing the role of Emma, **may be adaptive**, meaning it can change its behavior based on the results of your program's execution.

input
1 4 2 0 3

output

```
3

3

1

2
```

input

```
2 4
1 2 3 2
```

output

```

1 3
```

In the first example, there are four houses, and Emma lives in house number 1. Let a be the list of numbers written on the houses. Initially, $a = [0, 0, 0, 0]$, where 0 means that no number has been written on the corresponding house.

In the first run, it was determined that $k = 3$. Initially, you visited house number 2 and wrote the number 3 on it. Then you visited house number 0 and wrote the number 1 on it. After that, you visited house number 3 and wrote the number 2 on it. At this point, $a = [1, 0, 3, 2]$. In the end, Emma writes the number 2 on her house, after which $a = [1, 2, 3, 2]$.

In the second run, it was guessed that Emma lives in house 1 or in house 3. Since this guess is correct, the answer is valid, and Anna and Bertil win.

H. Records and Anti-records

3 seconds, 512 megabytes

A permutation of size n is a sequence of n integers $a_1, a_2, \ldots, a_n$, in which all values from 1 to n appear exactly once.

A sequence $b_1, b_2, \ldots, b_l$ is a subsequence of the sequence $a_1, a_2, \ldots, a_n$ if b can be obtained from a by removing some elements (that is, $l \le n$ and there exist $i_1 < i_2 < \ldots < i_l$ such that $a_{i_t} = b_t$).

- An element of the sequence a_i is called a *record* if it is strictly greater than all previous elements (that is, $a_j < a_i$ for all $1 \le j < i$).
- An element of the sequence a_i is called an *anti-record* if it is strictly less than all previous elements (that is, $a_j > a_i$ for all $1 \le j < i$).

Given a permutation $p_1, p_2, \ldots, p_n$ of size n, it is required to divide it into two non-empty subsequences q and r. In other words, each element of p must belong to exactly one of the subsequences. The goal is to maximize the sum of the number of records in q and the number of anti-records in r.

Input

Each test consists of several sets of input data. The first line contains a single integer t ($1 \le t \le 100\,000$) — the number of sets of input data. The following $2t$ lines contain the descriptions of the input data sets.

The first line of each input data set description contains a single integer n — the size of the permutation ($2 \le n \le 200\,000$).

The second line of each input data set description contains n integers $p_1, p_2, \ldots, p_n$ — the original permutation. It is guaranteed that among the elements of p, each number from 1 to n appears exactly once.

The sum of n across all input data sets does not exceed $200\,000$.

Output

For each input data set, output a single integer — the maximum possible sum of the number of records in q and the number of anti-records in r in the optimal partition.

input

```
4
5
4 1 2 3 5
10
3 8 10 4 1 2 7 9 5 6
3
1 2 3
6
4 2 5 1 6 3
```

output

```
5
6
3
5
```

One way to optimally split p into q and r in the first input data set (records in q and anti-records in r are highlighted):

- $q = \boxed{1}\ \boxed{2}\ \boxed{3}\ \boxed{5}$
- $r = \boxed{4}$

One way to optimally split p into q and r in the second input data set:

- $q = \boxed{3}\ \boxed{8}\ 4\ 1\ 2\ \boxed{9}$
- $r = \boxed{10}\ \boxed{7}\ \boxed{5}\ 6$

I. Traversals of a Binary Tree

2 seconds, 512 megabytes

A binary tree is a set of nodes, each of which may have a left and a right child. One of the nodes is the root of the tree, and it is not a child of any other node. Starting from the root and moving to one of the children each time, one can reach any node. The set of nodes that can be reached from a given node is called its subtree.

A binary tree has three main traversals: *pre-order*, *in-order*, and *post-order*.

The pre-order traversal of a tree is the order of its nodes obtained by the following recursive algorithm:

1. Add the root of the tree to the traversal.
2. If the root has a left child, write down the pre-order traversal of its subtree.
3. If the root has a right child, write down the pre-order traversal of its subtree.

In the in-order traversal, the root of the tree is written between the traversals of its children's subtrees, while in the post-order traversal, it is written after the traversals of its children's subtrees. In all variants of traversal, for each node, the left subtree is traversed first, followed by the right subtree.

Let's generalize these three variants of traversal: let each node store an integer x from -1 to 1, indicating when we write down this node, specifically:

- $x = -1$: before traversing the subtrees of its children;
- $x = 0$: between traversing the subtrees of its children;
- $x = 1$: after traversing the subtrees of its children.

Thus, if all nodes store -1, the traversal is pre-order; if 0, it is in-order; and if 1, it is post-order.

Consider a tree with n nodes numbered from 1 to n. The root of the tree is node 1. Initially, all nodes store the number -1.

As part of the research, it is necessary to process q queries of one of the following types:

1. Change the numbers in nodes $l, l + 1, \ldots, r$ to x (x is -1, 0, or 1).
2. Report the position of node i in the current traversal.

It is required to output answers to all queries of the second type.

Input
The first line of input contains two integers n and q ($1 \le n, q \le 100\,000$).

In the next n lines, two integers L_i and R_i ($0 \le L_i, R_i \le n$) are given for each node i — the number of the left and right child of node i, respectively, or 0 if the corresponding child is absent.

It is guaranteed that L_i and R_i define a valid binary tree.

In the following q lines, queries are given. The first number in the line is t ($t \in \{1, 2\}$) — the type of the query.

In the case of the first type of query, the following integers l, r, and x are given ($1 \le l \le r \le n$, x is -1, 0, or 1) — the boundaries of the segment of nodes where the numbers are changed, and the new value.

In the case of the second type of query, the number i ($1 \le i \le n$) is given — the number of the node for which the position in the traversal needs to be reported.

Output
For each query of the second type, output a single number from 1 to n — the position of the corresponding node in the traversal.

input
5 5 3 4 0 0 5 2 0 0 0 0 2 2 1 1 3 1 2 5 1 3 3 0 2 3
output
4 1 2

In the example, the traversal changes as follows:

- $[1, 3, 5, 2, 4]$
- $[5, 2, 3, 4, 1]$
- $[5, 3, 2, 4, 1]$

J. Sport is sport

1 second, 256 megabytes

As is known, Mike is an aspiring athlete. He follows a diet, never skips training, and also runs every morning at the stadium near his home.

The stadium where Mike runs is circular in shape, along which n trees are evenly spaced, numbered with integers from 1 to n. Each tree is characterized by its type. For convenience, we will number the types of trees with integers from 1 to m.

Mike's coach is very strict: he requires his students to provide photo reports of each training session, so Mike's run takes place in the following format. Mike stands in front of the first tree and starts running, taking pictures of all the trees he passes by. Unfortunately, Mike's phone has enough memory for only k photos. Therefore, after running past k trees, the boy is forced to stop, send some of the taken photos to his coach, clear the memory on his phone, and continue running. Mike understands that the coach will not be interested in looking at trees of the same type. Therefore, if during the run Mike takes several photos of trees of the same type, he will send only one of these photos. Then the boy clears the memory on his phone and continues running, starting from the next tree. This will continue until Mike stops again after taking k photos and discovers that the next tree is exactly the one from which he started running. After that, Mike will send the last batch of photos to his coach and go home.

To better understand the process, consider the following example: suppose there are six trees along the stadium of the following types: $1, 3, 2, 3, 2, 2$, and $k = 4$. Mike starts running and takes pictures of the first k trees, which have types 1, 3, 2, and 3. After this, Mike stops and sends the coach one photo of a tree of type 1, one photo of a tree of type 2, and one photo of a tree of type 3. After that, he clears the phone memory, stands in front of tree number 5, and continues running. Next, Mike will photograph trees of types 2, 2, 1, and 3, sending the coach one photo each of trees of the first, second, and third types. Then Mike will run past trees of types 2, 3, 2, and 2, and send the coach one photo of a tree of type 2 and one photo of a tree of type 3. After this, Mike will realize that the next tree is number 1 and will go home.

Thus, Mike will send the coach two photos of trees of the first type, three photos of trees of the second type, and three photos of trees of the third type.

One day, Mike became curious about how many photos the coach would see of each of the m types of trees. Help Mike answer this question. Calculate for each type of tree i the number of photos that will feature trees of that type.

Input
The first line contains three integers n, m, and k ($1 \le n \le 200\,000$, $1 \le m \le n$, $1 \le k \le n$) — the number of trees growing along the stadium, the number of different types of trees, and the maximum number of photos that Mike's phone memory can hold.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le m$) — the types of trees.

Output
Output m integers: the i-th number should equal the number of sent photos featuring trees of the i-th type.

input
4 2 2 2 1 2 2
output
1 2

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

input
7 3 5 2 1 2 3 1 3 3
output
7 7 7

Consider the first example. During the first run, Mike will take photos of trees numbered 1 and 2, and during the second — of trees numbered 3 and 4. After that, the run will end. The photos of trees of the second type will be sent by Mike twice, while the photo of the only tree of the first type will be sent only after the first run.

Consider the second example. During the first run, Mike will photograph trees numbered 1, 2, 3, and 4. During the second, trees numbered 5, 1, 2, and 3 will be captured on Mike's phone. During the third — trees numbered 4, 5, 1, and 2. During the fourth — trees numbered 3, 4, 5, and 1. Finally, during the fifth run, Mike will photograph trees numbered 2, 3, 4, and 5. Among any four consecutively planted trees, there is at least one tree of the first or second type, so they will be sent to the coach after each run. The tree of the third type will be sent after all runs except the third.

K. Conference

1 second, 512 megabytes

The Association of Scientific and Educational Communities is organizing a conference, during which it was planned to hold n events, numbered from 1 to n. The i-th event is defined by two integers l_i and r_i—the start and end times of the event.

Since some events may overlap or even completely coincide in time, one person may not always be able to attend all the events of the conference. We will consider that events i and j do not overlap if $r_i < l_j$ or $r_j < l_i$.

We will call a set of events *joint* if any two distinct events in this set do not overlap. Let the maximum size of the joint set of events at the conference be m. We will call the *density of the conference* the ratio $\frac{n}{m}$.

Due to budget cuts, the organizers of the conference have decided that the number of events at the conference will be reduced by exactly half. At the same time, they want to keep the density of the conference unchanged, so the maximum size of the joint set of selected events must also be reduced by exactly half. It turned out that in the original conference plan, both the number of events n and the maximum possible number of events in the joint set m are even numbers.

Help the organizers select a set of $\frac{n}{2}$ originally planned events to be held, so that the size of the maximum joint set of the selected events is equal to $\frac{m}{2}$.

Input

One test contains several sets of input data.

The first line contains one integer t—the number of sets of input data ($1 \le t \le 50,000$).

In the first line of each set description, there is one integer n—the number of events in the original plan ($2 \le n \le 100,000$, n is even).

In the next n lines of each set description, the events are described. In the i-th line, there are two integers l_i and r_i—the start and end times of the i-th event ($1 \le l_i < r_i \le 10^9$).

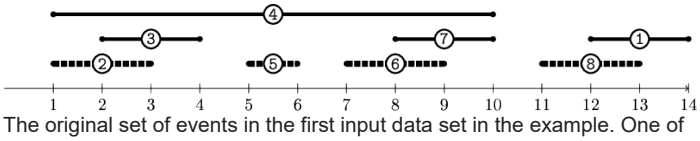
It is guaranteed that m—the size of the maximum joint set of events for the original plan is even.

Output

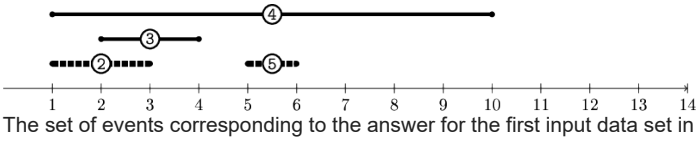
For each set of input data, output on a new line $\frac{n}{2}$ distinct event numbers that need to be held. If there are multiple suitable answers, you may output any of them. For the held events, the size of the maximum joint set of events must be equal to $\frac{m}{2}$.

input
2 8 12 14 1 3 2 4 1 10 5 6 7 9 8 10 11 13 6 1 2 2 4 1 2 1 4 5 7 6 8
output
2 5 3 4 1 2 3

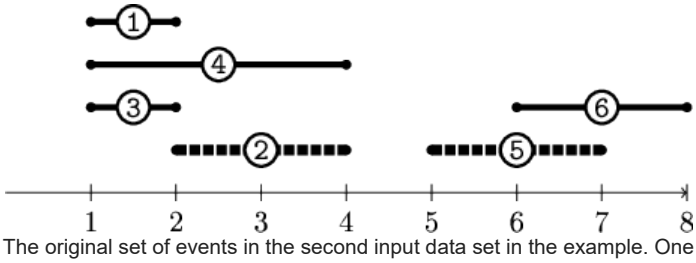
The drawings visualize the events. An event starting at time l_i and ending at time r_i is represented as a segment $[l_i, r_i]$.



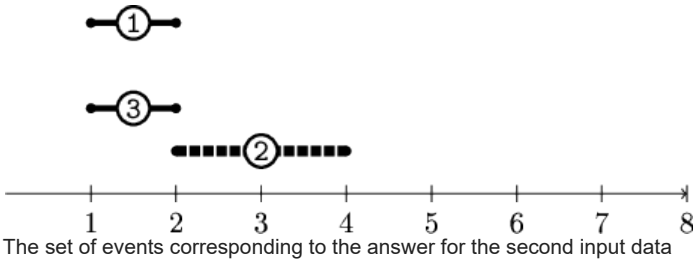
The original set of events in the first input data set in the example. One of the possible maximum joint sets is highlighted with a bold dashed line.



The set of events corresponding to the answer for the first input data set in the example. One of the possible maximum joint sets is highlighted with a bold dashed line.



The original set of events in the second input data set in the example. One of the possible maximum joint sets is highlighted with a bold dashed line.



The set of events corresponding to the answer for the second input data set in the example. One of the possible maximum joint sets is highlighted with a bold dashed line.

L. Apocalypse

3 seconds, 256 megabytes

On the planet Midav, the end of the world is very near. As is known, this flat planet can be represented as an infinite plane with Cartesian coordinates. There are Q settlements on this planet.

On day zero, an infection occurred on Midav. It is represented by a convex polygon with N vertices. Each day, the area of the infection changes in an unknown manner, but for each day with number $i > 0$, the following holds:

1. If any point on day i is infected at a distance d from the original polygon, then all other points at a distance not greater than d from the original polygon are also infected;
2. Let S_k be the area of the infection on day k. Then it holds that $S_i = 2 \cdot S_{i-1}$.

If any settlement is located inside or on the boundary of the infection, all living organisms in it will immediately die. For each settlement on the planet Midav, there is very little time left, so answer which day (including day zero) will be the last for the settlement.

Input

The first line contains an integer N — the number of points in the infection polygon on day zero $(3 \le N \le 10^5)$.

In the next N lines, two integers c_{xi} and c_{yi} are given — the coordinates of the vertices of the infection.

In the following line, there is an integer Q — the number of settlements on Midav $(1 \le Q \le 10^5)$.

In the next Q lines, two integers t_{xi} and t_{yi} are given — the coordinates of each settlement.

All coordinates do not exceed 10^9 in absolute value. It is guaranteed that the given polygon is convex and that the vertices are given in counter-clockwise order. It is guaranteed that the settlements are at least 10^{-6} away from the boundary of the infection on any day except day zero.

Output

Output Q integers — the last days for the settlements in the order of input.

input
4 1 3 1 1 3 1 3 3 4 2 2 1 2 4 1 6 2
output
0 0 2 4

In the example, the second settlement will be infected on day zero, as it lies on the boundary of the infection.

M. Maximizing Profit

1 second, 512 megabytes

Given a non-negative integer x consisting of n decimal digits. You can swap any two adjacent digits an arbitrary number of times. Each swap incurs a penalty of y. After performing the swaps, a bonus is awarded equal to the resulting number x_{new} obtained from x. Thus, if k swaps result in the number x_{new}, the profit is equal to $x_{new} - ky$.

We will call the number x_{new} optimal if it can be obtained from x through swaps, achieving the maximum possible profit.

Given x and y, determine the largest among the optimal numbers.

Input

The first line contains a single integer x, consisting of n decimal digits ($1 \le n \le 100\,000$). The number x may have leading zeros.

The second line contains a single integer y — the penalty for one digit swap ($1 \le y \le 10^{16}$).

Output

Output a single integer x_{new} — the largest among the optimal numbers. The number x_{new} must have a length of n and may contain leading zeros.

input
170 15
output
710

input
170 600
output
170

input
314599 17713
output
931459

input
001 1000
output
001

In the first example, after swapping the digits 1 and 7, the number 710 is obtained, and the profit is $710 - 15 = 695$.

In the second example, swapping the digits is not profitable; if the number is left as is, the profit is 170, while if swapped, the profit would be $710 - 600 = 110 < 170$.