

2026 Kruskal Cup Programming Contest



September 13, 2026

This problem set should contain 13 problems on 21 numbered pages.

**Prepared & Hosted by
University of Science and Technology of China Student
Computer Programming Club**





Problem A. Delete and Backspace

You are given an array a of length n . For each $k = 1, 2, \dots, n$, consider the following process independently. Initially, the array is a . You must perform exactly k operations. In each operation, you may choose one of the following two actions:

- **Backspace**: remove the first element of the current array;
- **Delete**: remove the last element of the current array.

Your score is defined as the value of the element removed in the k -th operation.

For each $k = 1, 2, \dots, n$, find the maximum score you can obtain.

Input

The first line contains an integer n , the length of the array ($1 \leq n \leq 10^5$).

The second line contains n integers $a_1, a_2, \dots, a_n$, representing the array a ($1 \leq a_i \leq 10^9$).

Output

Output n integers. The k -th integer should be the maximum score you can obtain after performing exactly k operations.

Example

standard input	standard output
5 2 7 8 1 4	4 7 8 8 8

Note

For $k = 1$, we can only delete either the leftmost element 2 or the rightmost element 4, so the maximum score is 4.

For $k = 2$, we can first delete the leftmost element 2, and then delete the new leftmost element 7. Thus, the element deleted in the second operation is 7, so the maximum score is 7.

For $k = 3$, we can delete the leftmost elements 2, 7, 8 in order, making the element deleted in the third operation equal to 8. Therefore, the maximum score is 8.

For $k = 4$ and $k = 5$, we can similarly arrange the previous operations so that the element deleted in the last operation is 8.



Problem B. Vanishing Inversions

Alice and Bob are playing a game on a permutation, with Alice moving first.

A sequence of length n is called a **permutation** if and only if each integer from $1, 2, \dots, n$ appears exactly once in the sequence.

Initially, a permutation $a_1, a_2, \dots, a_n$ of length n is given. At any moment during the game, the current sequence is always a permutation of some length m . The players take turns, and on each turn the current player must choose one of the following two operations:

- Choose an index i ($1 \leq i < m$) such that $a_i > a_{i+1}$, and swap a_i and a_{i+1} ;
- Choose an index i ($1 \leq i < m$) such that $a_{i+1} = a_i + 1$, delete a_i and a_{i+1} , and then renumber the remaining elements according to their relative order: the k -th smallest remaining element is renumbered as k , so that the remaining sequence becomes a permutation again.

For example, suppose that the current permutation is $[2, 3, 4, 1]$. The adjacent elements 2 and 3 can be deleted, leaving $[4, 1]$; after renumbering, the permutation becomes $[2, 1]$.

A player who cannot perform any operation loses the game. Assuming that both Alice and Bob play optimally, determine the winner.

Input

Each test contains multiple test cases.

The first line contains an integer T ($1 \leq T \leq 5000$), the number of test cases.

For each test case:

- The first line contains an integer n ($1 \leq n \leq 5000$).
- The second line contains n integers $a_1, a_2, \dots, a_n$. It is guaranteed that a is a permutation of length n .

It is guaranteed that the sum of n over all test cases does not exceed 5000.

Output

For each test case, output **Alice** if Alice wins; otherwise, output **Bob**.

Example

standard input	standard output
5	Bob
1	Alice
1	Bob
2	Alice
1 2	Bob
2	
2 1	
4	
2 4 1 3	
5	
5 1 4 2 3	



Problem C. Range Minimization

The **range** of an array is defined as the difference between its maximum element and its minimum element. You are given an integer array a of length n . You may perform the following operation any number of times, including zero times:

- Choose two distinct indices i, j ($1 \leq i, j \leq n$ and $i \neq j$), and update the array as follows:

$$a_i \leftarrow a_i + 3, \quad a_j \leftarrow a_j - 1.$$

Calculate the minimum possible range of a after performing any finite number of operations.

Input

Each test contains multiple test cases.

The first line contains an integer T ($1 \leq T \leq 10^4$), the number of test cases.

For each test case:

- The first line contains an integer n ($1 \leq n \leq 2 \times 10^5$), the length of the array a ;
- The second line contains n integers $a_1, a_2, \dots, a_n$ ($-10^9 \leq a_i \leq 10^9$), the elements of a .

It is guaranteed that the sum of n over all test cases does not exceed 2×10^5 .

Output

For each test case, output one integer on a separate line, denoting the minimum possible range after any finite number of operations.

Example

standard input	standard output
3	0
1	2
5	1
2	
0 2	
4	
-1 9 5 8	

Note

In the first test case, the array contains only one element, so its range is always 0.



Problem D. Corridor Tuning

There are n resonators placed in order along a corridor. The current frequency offset of the i -th resonator is a_i , and its target offset is b_i .

The tuning device can only be activated from one end of the corridor. In one operation, you may choose any nonzero integer x and perform one of the following operations:

1. Choose k ($1 \leq k \leq n$) and add x to every number in the prefix $a_1, a_2, \dots, a_k$;
2. Choose k ($1 \leq k \leq n$) and add x to every number in the suffix $a_k, a_{k+1}, \dots, a_n$.

x may be positive or negative. Regardless of the magnitude of $|x|$, the modification counts as exactly one operation.

Find the minimum number of operations required to transform sequence a into sequence b .

Input

The first line contains an integer T ($1 \leq T \leq 10^4$), the number of test cases.

For each test case:

- The first line contains an integer n ($1 \leq n \leq 2 \times 10^5$).
- The second line contains n integers $a_1, a_2, \dots, a_n$ ($-10^9 \leq a_i \leq 10^9$).
- The third line contains n integers $b_1, b_2, \dots, b_n$ ($-10^9 \leq b_i \leq 10^9$).

It is guaranteed that the sum of n over all test cases does not exceed 2×10^5 , and the sums of $\sum |a_i - a_{i-1}|$ ($i \geq 2$) and $\sum |b_i - b_{i-1}|$ ($i \geq 2$), taken over all test cases, do not exceed 10^4 respectively.

Output

For each test case, output one integer on a separate line, the minimum number of operations.

Example

standard input	standard output
4	2
3	3
0 0 0	1
-1 2 0	0
3	
0 0 0	
3 1 4	
4	
1 4 2 8	
6 9 7 13	
2	
-7 10	
-7 10	

Note

For the first test case, one possible sequence of operations is:

1. Add -3 to a prefix of length 1, obtaining $[-3, 0, 0]$.
2. Add 2 to a prefix of length 2, obtaining $[-1, 2, 0]$.



For the third test case, adding 5 to the entire sequence is sufficient.



Problem E. Punctured Neighborhood

This is an interactive problem.

There is an unknown sequence B consisting of 0s and 1s, with 1-based indexing. You can make multiple queries. In each query, you specify a center m and a radius r , and the interactor will return the number of 1s in the punctured neighborhood of indices $[m - r, m + r] \setminus \{m\}$ of B . Try to determine the total number of 1s in B .

Note that the interval you query must not exceed the boundaries of the sequence itself.

Interaction Protocol

There are multiple test cases in this problem.

First, read a line containing a single positive integer T ($1 \leq T \leq 1000$), denoting the number of test cases.

For each test case:

- First read a line containing a single integer n ($4 \leq n \leq 1000$), denoting the length of B .
- To make a query, output a line in the format `? m r` and **flush the output buffer**. It is required that $2 \leq m \leq n - 1, r \geq 1, m - r \geq 1, m + r \leq n$. Then, read a line containing a non-negative integer, denoting the value of $\sum_{i=1}^r (B_{m-i} + B_{m+i})$.
- When you have found the answer, output a line in the format `! x` and **flush the output buffer**, where x represents the total number of 1s in B .
- As an exception, if it is impossible to determine the answer no matter how queries are made, output a line `! -1` and **flush the output buffer**.

The number of queries for each test case must not exceed 35.

Example

standard input	standard output
2	
4	
	? 2 1
1	
	? 3 1
1	
	! 2
5	
	? 3 2
4	
	? 2 1
2	
	! 5

Note

The example demonstrates a possible interaction process. The unknown sequences in the two test cases are 1100 and 11111, respectively.

How to **flush the output buffer**:

- In C and C++, use `fflush(stdout)` (if using `printf`) or `cout.flush()` (if using `cout`); Using `cout << endl` will also flush the buffer in C++.
- In Python, use `stdout.flush()`;



Problem F. A Simple Construction Problem

For a positive integer n , a sequence of length n is called a “red-brick sequence” if it satisfies the following conditions:

- Each of $1, 2, \dots, n$ appears exactly once in the sequence.
- The sum of every pair of adjacent elements is not a perfect square.

Given an integer n , construct a red-brick sequence of length n .

Input

The first line contains an integer T ($1 \leq T \leq 10^4$), the number of test cases.

For each test case, the only line contains an integer n ($1 \leq n \leq 4 \times 10^6$), the required length of the red-brick sequence.

It is guaranteed that the sum of n does not exceed 4×10^6 . It can be proved that a solution always exists under the given constraints.

Output

Output T lines, each containing the sequence to be constructed. If there are multiple answers, output any of them.

Example

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

Note

For $n = 5$, if the output sequence is 1 2 3 4 5, it is a wrong answer because $4 + 5 = 9$ is a perfect square.



Problem G. Nailoong vs. Bombloong 3.1

Input file: standard input
Output file: standard output

This is a communication problem. In this problem, your program will run twice. Between the two runs, all variables stored in memory will be lost, but the information obtained during the first run may be crucial for solving the problem during the second run.

There are two characters in this problem: “Nailoong” and “Bombloong”.

Nailoong has complete structural information of a tree with n nodes¹, while Bombloong only knows the number of nodes n . Since Bombloong has been imprisoned by the evil little leopard, Nailoong can only help Bombloong reconstruct a tree that is **isomorphic**² to the original tree through a special one-way communication method.

The rules of communication are as follows:

- Nailoong must color every vertex of the tree **either black, white, or grey**;
- the little leopard generates a sequence according to the code below;
- Nailoong can send only the sequence $c_1, c_2, \dots, c_{2n-1}$ of length $2n - 1$ to Bombloong.

Algorithm 1 Encode($T, color$)

```

1: procedure DFS( $u, parent$ )
2:   append  $u$  to  $tour$ 
3:   for  $v$  in vertices adjacent to  $u$  do                                 $\triangleright v$  can be iterated in any order.
4:     if  $v \neq parent$  then
5:       DFS( $v, u$ )
6:       append  $u$  to  $tour$ 
7:     end if
8:   end for
9: end procedure
10: procedure GENERATESEQUENCE( $T, color$ )
11:    $tour \leftarrow$  empty sequence
12:    $root \leftarrow$  the starting vertex chosen by the leopard
13:   DFS( $root, NIL$ )
14:    $c \leftarrow$  empty sequence
15:   for  $v$  in  $tour$  do
16:     append  $color_v$  to  $c$ 
17:   end for
18:   return  $c$ 
19: end procedure

```

Nailoong cannot directly send the node indices to Bombloong; she can only send the sequence c . After receiving n and the sequence c , Bombloong must construct and output a tree **isomorphic** to Nailoong’s tree.

¹Tree : a tree with n nodes can be represented by $T(V, E)$, where $|V| = n$ and $|E| = n - 1$. $E \subseteq \{\{u, v\} | u, v \in V\}$, and $\forall u, v \in V, u \neq v, \exists u = p_0, p_1, \dots, p_k = v, \{p_{i-1}, p_i\} \in E$ holds for all $i = 1, 2, \dots, k$. The elements in V are nodes and the elements in E are edges. Node u and node v are adjacent if and only if $\{u, v\} \in E$.

²Isomorphism: Two trees $T_1(V_1, E_1)$ and $T_2(V_2, E_2)$ are isomorphic if and only if there exists a bijection $f : V_1 \rightarrow V_2$ such that $\forall \{u, v\} \in E_1, \{f(u), f(v)\} \in E_2$.



Communication Protocol

In each test, the contestant's program will be run twice. A testing tool is provided in the handout files for local debugging.

First Run

In the first run, you play the role of "Nailoong".

Input

The first line contains a string **first**, which allows your program to identify that this is the first run.

The second line contains an integer T ($1 \leq T \leq 10^4$), the number of testcases.

For each testcase:

- The first line contains an integer n ($2 \leq n \leq 2 \times 10^5$), the number of nodes in the tree.
- The next $n - 1$ lines each contain two integers u, v ($1 \leq u, v \leq n$), denoting an edge of the tree.

It is guaranteed that the sum of n over all testcases does not exceed 2×10^5 .

Output

Your output should contain T lines. The i -th line contains a sequence $col_1, col_2, \dots, col_n$ of length n , where each $col_i \in \{0, 1, 2\}$ ($i = 1, 2, \dots, n$), denoting the color assigned by Nailoong to the i -th vertex. $col_i = 0$ denotes white, $col_i = 1$ denotes black, and $col_i = 2$ denotes grey.

Second Run

In the second run, you play the role of "Bombloong".

Input

The first line contains a string **second**, which allows your program to identify that this is the second run.

The second line contains an integer T ($1 \leq T \leq 10^4$), the number of testcases.

For each testcase:

- The first line contains an integer n ($2 \leq n \leq 2 \times 10^5$), the number of nodes in the tree.
- The next line contains $2n - 1$ integers, denoting the sequence generated by the judge according to Nailoong's coloring.

It is guaranteed that the sum of n over all testcases does not exceed 2×10^5 .

Output

For the i -th test case, let the input number of nodes be n . Output $n - 1$ lines, each containing two integers u, v , denoting an edge ($1 \leq u, v \leq n$) of the reconstructed tree. You must ensure that the output tree is isomorphic to the tree given in the first run.



Examples

standard input	standard output
first 2 2 1 2 3 1 2 2 3	0 1 0 1 2
second 2 3 0 1 2 1 0 2 0 1 0	1 3 2 3 1 2

Note

The two samples demonstrate the two runs of the same test point.

Note that if the trees in the first input are $t_1, \dots, t_T$ in order, and the output color sequences are $col_1, \dots, col_T$, let the evaluator obtain color sequences a_i from t_i, col_i ($1 \leq i \leq T$). The evaluator then randomly generates a permutation $p_1, \dots, p_T$ of $1 \sim T$, sends $a_{p_1}, \dots, a_{p_T}$ to the second input, and the output trees are $t'_1, t'_2, \dots, t'_T$. The evaluator compares t_{p_i} with t'_i .

How to Use the Handout Files

```
python3 tree_communication_testing_tool1.py data.in ./solution
python3 tree_communication_testing_tool1.py --trials 10 data.in ./solution
python3 tree_communication_testing_tool1.py data.in python3 solution.py
```

Here, `solution` and `solution.py` are a C/C++ executable and a Python source file, respectively. The file `data.in` is the first-run input; `trials` is the number of repeated test times.



Problem H. Nailoong vs. Bombloong 3.2

Input file: standard input
Output file: standard output

This is a communication problem. In this problem, your program will run twice. Between the two runs, all variables stored in memory will be lost, but the information obtained during the first run may be crucial for solving the problem during the second run.

There are two characters in this problem: “Nailoong” and “Bombloong”.

Nailoong has complete structural information of a tree with n nodes¹, while Bombloong only knows the number of nodes n . Since Bombloong has been imprisoned by the evil little leopard, Nailoong can only help Bombloong reconstruct a tree that is **isomorphic**² to the original tree through a special one-way communication method.

The rules of communication are as follows:

- Nailoong must color every vertex of the tree **either black or white**;
- the little leopard generates a sequence according to the code below;
- Nailoong can send only the sequence $c_1, c_2, \dots, c_{2n-1}$ of length $2n - 1$ to Bombloong.

Algorithm 2 Encode($T, color$)

```

1: procedure DFS( $u, parent$ )
2:   append  $u$  to  $tour$ 
3:   for  $v$  in vertices adjacent to  $u$  do                                 $\triangleright v$  can be iterated in any order.
4:     if  $v \neq parent$  then
5:       DFS( $v, u$ )
6:       append  $u$  to  $tour$ 
7:     end if
8:   end for
9: end procedure
10: procedure GENERATESEQUENCE( $T, color$ )
11:    $tour \leftarrow$  empty sequence
12:    $root \leftarrow$  the starting vertex chosen by the leopard
13:   DFS( $root, NIL$ )
14:    $c \leftarrow$  empty sequence
15:   for  $v$  in  $tour$  do
16:     append  $color_v$  to  $c$ 
17:   end for
18:   return  $c$ 
19: end procedure

```

Nailoong cannot directly send the node indices to Bombloong; she can only send the sequence c . After receiving n and the sequence c , Bombloong must construct and output a tree **isomorphic** to Nailoong’s tree.

¹Tree : a tree with n nodes can be represented by $T(V, E)$, where $|V| = n$ and $|E| = n - 1$. $E \subseteq \{\{u, v\} | u, v \in V\}$, and $\forall u, v \in V, u \neq v, \exists u = p_0, p_1, \dots, p_k = v, \{p_{i-1}, p_i\} \in E$ holds for all $i = 1, 2, \dots, k$. The elements in V are nodes and the elements in E are edges. Node u and node v are adjacent if and only if $\{u, v\} \in E$.

²Isomorphism: Two trees $T_1(V_1, E_1)$ and $T_2(V_2, E_2)$ are isomorphic if and only if there exists a bijection $f : V_1 \rightarrow V_2$ such that $\forall \{u, v\} \in E_1, \{f(u), f(v)\} \in E_2$.



Communication Protocol

In each test, the contestant's program will be run twice. A testing tool is provided in the handout files for local debugging.

First Run

In the first run, you play the role of "Nailoong".

Input

The first line contains a string **first**, which allows your program to identify that this is the first run.

The second line contains an integer T ($1 \leq T \leq 10^4$), the number of testcases.

For each testcase:

- the first line contains an integer n ($2 \leq n \leq 2 \times 10^5$), the number of nodes in the tree.
- The next $n - 1$ lines each contain two integers u, v ($1 \leq u, v \leq n$), denoting an edge of the tree.

It is guaranteed that the sum of n over all testcases does not exceed 2×10^5 .

Output

Your output should contain T lines. The i -th line contains a sequence $col_1, col_2, \dots, col_n$ of length n , where each $col_i \in \{0, 1\}$ ($i = 1, 2, \dots, n$), denoting the color assigned by Nailoong to the i -th vertex. $col_i = 0$ denotes white, and $col_i = 1$ denotes black.

Second Run

In the second run, you play the role of "Bombloong".

Input

The first line contains a string **second**, which allows your program to identify that this is the second run.

The second line contains an integer T ($1 \leq T \leq 10^4$), the number of testcases.

For each testcase:

- The first line contains an integer n ($2 \leq n \leq 2 \times 10^5$), the number of nodes in the tree.
- The next line contains $2n - 1$ integers, denoting the sequence generated by the judge according to Nailoong's coloring.

It is guaranteed that the sum of n over all testcases does not exceed 2×10^5 .

Output

For the i -th test case, let the input number of nodes be n . Output $n - 1$ lines, each containing two integers u, v , denoting an edge ($1 \leq u, v \leq n$) of the reconstructed tree. You must ensure that the output tree is isomorphic to the tree given in the first run.



Examples

standard input	standard output
first 2 2 1 2 3 1 2 2 3	0 1 0 1 0
second 2 3 0 1 0 1 0 2 0 1 0	1 3 2 3 1 2

Note

The two samples demonstrate the two runs of the same test point.

Note that if the trees in the first input are $t_1, \dots, t_T$ in order, and the output color sequences are $col_1, \dots, col_T$, let the evaluator obtain color sequences a_i from t_i, col_i ($1 \leq i \leq T$). The evaluator then randomly generates a permutation $p_1, \dots, p_T$ of $1 \sim T$, sends $a_{p_1}, \dots, a_{p_T}$ to the second input, and the output trees are $t'_1, t'_2, \dots, t'_T$. The evaluator compares t_{p_i} with t'_i .

How to Use the Handout Files

```
python3 tree_communication_testing_tool2.py data.in ./solution
python3 tree_communication_testing_tool2.py --trials 10 data.in ./solution
python3 tree_communication_testing_tool2.py data.in python3 solution.py
```

Here, `solution` and `solution.py` are a C/C++ executable and a Python source file, respectively. The file `data.in` is the first-run input; `trials` is the number of repeated test times.



Problem I. Same Frequency Echoes

You are given a weighted tree with n nodes¹, rooted at node 1.

Node i has:

- a frequency band c_i ;
- a transmission time a_i .

Each frequency band c has an importance w_c .

For two distinct nodes u, v using the same frequency band, define their **synchronization cost** as

$$D(u, v) = |a_u - a_v| + \text{dist}(u, v),$$

where $\text{dist}(u, v)$ is the sum of edge weights on the simple path between u and v ².

Each query gives a node x and a nonnegative integer K .

We say that frequency band c produces an echo in the jurisdiction of x if and only if there exist two distinct nodes

$$u, v \in \text{subtree}(x)$$

such that

$$c_u = c_v = c, \quad D(u, v) \leq K.$$

Here, $\text{subtree}(x)$ denotes the subtree rooted at x ³.

For each query, calculate the sum of the importance values of all frequency bands that produce an echo, namely

$$\sum_{c=1}^m w_c [\exists u \neq v \in \text{subtree}(x), c_u = c_v = c, D(u, v) \leq K].$$

The brackets denote the Iverson bracket: its value is 1 if the condition holds, and 0 otherwise.

Input

The first line contains three integers n, m, q ($1 \leq n \leq 10^6$, $1 \leq m \leq 10^5$, $1 \leq q \leq 10^6$), denoting the number of nodes, the number of frequency bands, and the number of queries, respectively.

The second line contains m integers $w_1, w_2, \dots, w_m$ ($0 \leq w_c \leq 10^9$), where w_c is the importance of frequency band c .

The third line contains n integers $c_1, c_2, \dots, c_n$ ($1 \leq c_i \leq m$), where c_i is the frequency band used by node i .

The fourth line contains n integers $a_1, a_2, \dots, a_n$ ($0 \leq a_i \leq 10^9$), where a_i is the transmission time of node i .

Each of the next $n - 1$ lines contains three integers u, v, d ($1 \leq u, v \leq n$, $0 \leq d \leq 10^9$), indicating that there is an undirected edge of weight d between nodes u and v .

Each of the next q lines contains two integers x, K ($1 \leq x \leq n$, $0 \leq K \leq 4 \times 10^{18}$), describing a query.

It is guaranteed that the input edges form a tree and that every answer fits in a signed 64-bit integer.

Output

For each query, output one integer on a separate line, denoting the answer.

¹Tree : a tree with n nodes can be represented by $T(V, E)$, where $|V| = n$ and $|E| = n - 1$. $E \subseteq \{\{u, v\} | u, v \in V\}$, and $\forall u, v \in V, u \neq v, \exists u = p_0, p_1, \dots, p_k = v, \{p_{i-1}, p_i\} \in E$ holds for all $i = 1, 2, \dots, k$. The elements in V are nodes and the elements in E are edges. Node u and node v are adjacent if and only if $\{u, v\} \in E$.

²Here the simple path between u and v is $u = p_0, \dots, p_k = v$, where $\forall i \neq j, p_i \neq p_j$ and $\{p_{i-1}, p_i\} \in E$ holds $\forall i = 1, 2, \dots, k$.

³ v is in the subtree rooted at u if and only if u is on the simple path between the root and the node v .



Example

standard input	standard output
7 3 5	0
5 7 11	12
1 2 1 3 2 1 3	7
10 4 13 8 9 20 7	5
1 2 2	23
1 3 3	
2 4 4	
2 5 1	
3 6 2	
3 7 5	
1 5	
1 6	
2 6	
3 9	
1 15	

Note

The synchronization costs between nodes using the same frequency band are as follows:

- The nodes using frequency band 1 are 1, 3, 6: $D(1, 3) = |10 - 13| + 3 = 6$, $D(1, 6) = |10 - 20| + 5 = 15$, and $D(3, 6) = |13 - 20| + 2 = 9$;
- The nodes using frequency band 2 are 2, 5: $D(2, 5) = |4 - 9| + 1 = 6$;
- The nodes using frequency band 3 are 4, 7: $D(4, 7) = |8 - 7| + 14 = 15$.

Therefore:

- In query (1, 5), no frequency band produces an echo, so the answer is 0;
- In query (1, 6), frequency bands 1, 2 produce echoes, so the answer is $5 + 7 = 12$;
- In query (2, 6), only frequency band 2 produces an echo, so the answer is 7;
- In query (3, 9), only frequency band 1 produces an echo, so the answer is 5;
- In query (1, 15), all three frequency bands produce echoes, so the answer is $5 + 7 + 11 = 23$.



Problem J. Emordnilap

Given a 1-indexed string s of length n . For each $1 \leq i \leq n$, a non-negative integer r is called a palindromic radius of center i if the substring of length $2r + 1$ centered at i is a palindrome.

For each center i , find the palindromic radius such that the corresponding palindromic substring is lexicographically smallest.

Input

There are multiple test cases.

The first line contains a single integer T ($1 \leq T \leq 10^6$), denoting the number of test cases.

For each test case, there is a single line containing a non-empty string s . It is guaranteed that s consists only of lowercase English letters.

It is guaranteed that $\sum |s| \leq 10^6$.

Output

Output T lines. For each test case, output a line containing $|s|$ integers, where the i -th integer represents the answer for center i .

Example

standard input	standard output
2	0 0 1 0 0 0 0
cabacbc	0
a	

Note

In the first sample test case, there are three palindromic substrings centered at 3, namely **b**, **aba**, and **cabac**. The lexicographically smallest among them is **aba**, with a radius of 1.



Problem K. Card Packs

You are given two coprime positive integers p and q . There are pq different types of cards, numbered $0, 1, \dots, pq - 1$.

There are two types of standard card packs:

- Type-1 standard card packs: there are p kinds, numbered by u ($0 \leq u < p$). The pack numbered u contains one copy of every card whose number satisfies $x \equiv u \pmod{p}$, so each pack contains exactly q cards.
- Type-2 standard card packs: there are q kinds, numbered by v ($0 \leq v < q$). The pack numbered v contains one copy of every card whose number satisfies $x \equiv v \pmod{q}$, so each pack contains exactly p cards.

There are now n cards arranged on the table, where the number of the i -th card is a_i . You need to process Q operations of the following two types:

- 1 i x: replace the number of the i -th card with x ;
- 2 l r: determine whether all cards in the interval $[l, r]$ can be partitioned exactly into several standard card packs.

Note: When repacking the cards, every card in the interval must belong to exactly one pack, and each kind of standard card pack may be used any number of times.

Input

The first line contains four integers n, p, q, Q ($1 \leq n, p, q, Q \leq 3 \times 10^5$, $2 \leq p + q \leq 3 \times 10^5$, $\gcd(p, q) = 1$). The second line contains n integers $a_1, a_2, \dots, a_n$ ($0 \leq a_i < pq$), denoting the initial number of each card. The next Q lines each describe an operation in the form 1 i x or 2 l r:

- For every operation of the first type, it is guaranteed that $1 \leq i \leq n$ and $0 \leq x < pq$.
- For every operation of the second type, it is guaranteed that $1 \leq l \leq r \leq n$.

Output

For each operation of the second type, if the specified cards can be partitioned exactly into several standard card packs, output YES; otherwise, output NO.

Example

standard input	standard output
8 2 3 5	YES
0 0 2 3 4 1 4 2	NO
2 1 5	NO
2 6 8	YES
1 8 5	
2 6 8	
2 6 7	

Note

When $p = 2, q = 3$:

- There are two Type-1 standard card packs: $\{0, 2, 4\}$ and $\{1, 3, 5\}$.



- There are three Type-2 standard card packs: $\{0, 3\}$, $\{1, 4\}$, and $\{2, 5\}$.

For the first query, the multiset of cards in the interval is $\{0, 0, 2, 3, 4\}$. It can be partitioned into $\{0, 2, 4\} + \{0, 3\}$, so the answer is **YES**.

For the last query, the multiset of cards in the interval is $\{1, 4\}$. It forms exactly one Type-2 standard card pack, so the answer is **YES**.



Problem L. AC Automaton

Given a deterministic finite automaton (DFA) with a state set $Q = \{1, 2, \dots, n\}$, a start state $q = 1$, an accept state set F of size m , and an alphabet $\Sigma = \{\mathbf{A}, \mathbf{C}\}$. Find the lexicographically smallest string accepted by the automaton with a length of at least k . Since the answer might be very long, if the length of the answer exceeds n , you only need to output the last n characters of the answer. If no string satisfies the aforementioned requirements, output **WA**.

A string is accepted by the automaton if and only if there exists a walk starting from the initial state and ending at an accept state, such that the letters on the traversed transition edges exactly form the string. The walk can pass through repeated vertices and edges.

Input

There are multiple test cases.

The first line contains a single integer T ($1 \leq T \leq 5 \times 10^5$), denoting the number of test cases.

For each test case:

- The first line contains three space-separated integers: the number of states n ($2 \leq n \leq 10^6$), the number of accept states m ($1 \leq m \leq n - 1$), and the lower bound of the length k ($1 \leq k \leq 10^9$).
- The next line contains m space-separated distinct integers, denoting the set of accept states. It is guaranteed that these numbers are in the range $[2, n]$.
- Then n lines follow, each containing 2 space-separated integers. The two integers on the i -th line denote the states transitioned to from state i along **A** and **C**, respectively.

It is guaranteed that $\sum n \leq 10^6$.

Output

Output T lines, each containing a string, where the i -th line represents the answer for the i -th test case.

Example

standard input	standard output
2	CCCCA
5 1 6	WA
4	
2 5	
5 3	
4 2	
5 5	
5 5	
3 1 2	
2	
2 3	
3 3	
3 3	

Note

In the first test case of the sample, the language recognized by the DFA is $\{\mathbf{AC}^{2t+1}\mathbf{A} \mid t \geq 0\}$.



Problem M. Iterated Fixed Points

You are given an integer n , an integer k , and an integer p .

Consider all functions

$$f : \{1, 2, \dots, n\} \rightarrow \{1, 2, \dots, n\}.$$

Let f^k denote the k -fold iteration of f . If $x \in \{1, 2, \dots, n\}$ satisfies $f^k(x) = x$, then x is called a fixed point of f^k .

Find the number of functions f that have exactly p fixed points under f^k . Output the answer modulo $10^9 + 7$.

Input

Each test contains multiple test cases.

The first line contains an integer T ($1 \leq T \leq 10^4$), the number of test cases.

Each of the next T lines contains three integers n, k, p ($1 \leq n, k \leq 10^6$, $0 \leq p \leq n$).

It is guaranteed that the sum of n over all test cases does not exceed 10^6 .

Output

For each test case, output one integer on a separate line, denoting the answer modulo $10^9 + 7$.

Example

standard input	standard output
7	12
3 2 2	1
2 1 0	8
3 1 0	2
3 2 0	3
3 3 3	10
4 2 4	54
4 1 2	