Problem A. Cipher

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	1024 megabytes

Alice designed a cipher system: it encodes the plaintext as a 64-bit unsigned integer x, and randomly selects a 64-bit unsigned integer a as the public key, encrypting the plaintext to get ciphertext $b = a^x \bmod 2^{64}$. In particular, it is defined that $0^0 = 1$.

Now, Bob has intercepted n sets of encrypted information (a_i, b_i) $(i = 1, 2, \cdots, n)$. To decrypt all the information, Bob needs to find the **smallest** integer x_i $(0 \le x_i < 2^{64})$ for each set of encrypted information such that ${a_i}^{x_i} \equiv b_i \pmod{2^{64}}$, or determine that the information has been corrupted (i.e., there is no x_i that satisfies the condition). Please write a program to help Bob accomplish this task.

Input

The first line of input contains an integer n $(1 \le n \le 10^5)$, representing the number of messages.

The next n lines, the i-th line contains two integers a_i and b_i $(0 \le a_i, b_i < 2^{64})$, representing the public key and ciphertext of the i-th set of information, respectively.

Output

Output n lines, for the i-th line:

- If there exists at least one x_i that satisfies the conditions stated in the problem, output the smallest one;
- Otherwise, output a line `broken message`.

Example

standard input	standard output
5 2 4 3 27 1000000007 998244353 4 2 1000000007 1000000009	2 3 994996658310742016 broken message broken message

Problem B. Fortress

Input file:	`standard input`
Output file:	`standard output`
Time limit:	6 seconds
Memory limit:	1024 megabytes

In a distant land, there is a fortress. If we represent the orientation using a Cartesian coordinate system, the territory of the fortress can be viewed as a circle centered at the origin with radius R. The boundary of the fortress territory is surrounded by high walls, and it is impossible to see the outside from within the fortress.

Inside the fortress, there are some buildings. For convenience, the boundaries of the buildings are represented by n **possibly overlapping** rectangles whose edges are parallel to the coordinate axes. A point inside the fortress is considered blocked if it strictly lies within any of the rectangles (points on the boundary of the rectangles are not considered to be inside).

Little S is a guard of this fortress. He stands at some unblocked point (x, y) inside the fortress. Little S can guard another point (x', y') inside the fortress if and only if every point on the line segment $(x, y) - (x', y')$ (including the two endpoints) is not blocked. The task is to calculate the area of the set of points that Little S can guard, which represents the area of the fortress land that Little S can oversee.

Input

This problem contains multiple test cases. The first line of input contains an integer T ($1 \le T \le 10^5$), indicating the number of test cases.

For each test case:

The first line contains two integers R, n ($0 < R \le 10^6, 0 \le n \le 10^5$), representing the radius of the fortress and the number of rectangles.

The next n lines each contain four integers x_1, y_1, x_2, y_2 ($|x_1|, |x_2|, |y_1|, |y_2| \le 10^6, x_1 < x_2, y_1 < y_2$), describing a rectangle, where (x_1, y_1) represents the coordinates of the bottom-left corner and (x_2, y_2) represents the coordinates of the top-right corner. It is guaranteed that all rectangles are strictly within the fortress territory and do not touch the boundary of the territory.

The last line contains two integers x, y ($|x|, |y| \le 10^6, x^2 + y^2 < R^2$), representing the coordinates of Little S. It is guaranteed that the point (x, y) is not blocked.

It is guaranteed that the total sum of n across all test cases does not exceed 10^5.

Output

For each test case, output a single line with a real number representing the area of the set of points that Little S can guard within the fortress.

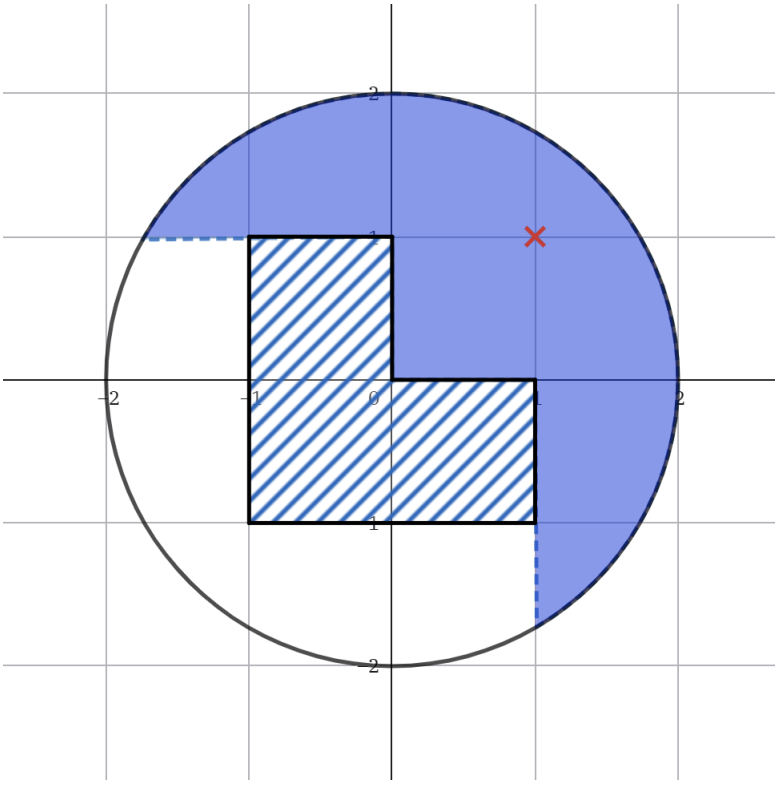
The answer will be considered correct if the relative or absolute error compared to the standard answer is less than 10^{-6}. In other words, if your output is a and the standard answer is b, your output is considered correct if and only if $\frac{|a-b|}{\max(1,b)} \le 10^{-6}$.

Example

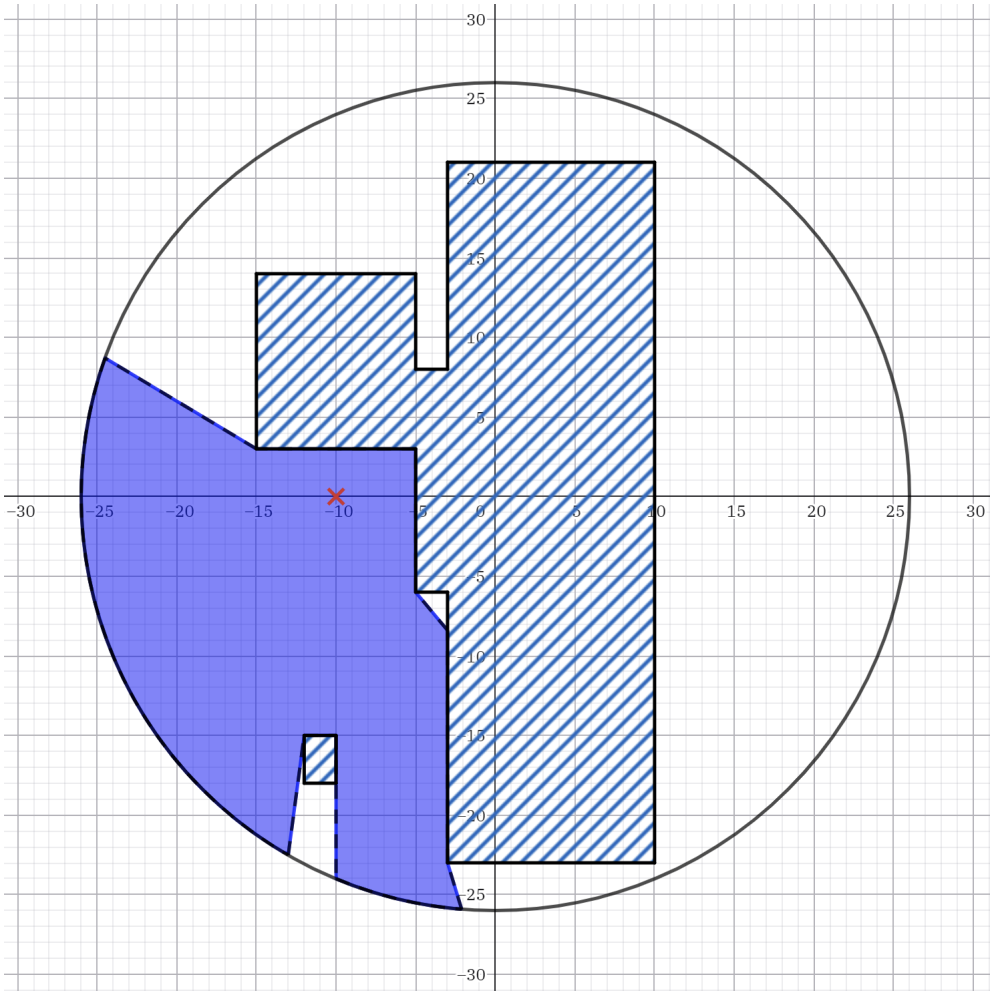
standard input	standard output
2 2 3 -1 -1 0 0 -1 -1 0 1 0 -1 1 0 1 1 26 5 -3 -23 10 21 -15 3 -5 14 -12 -18 -10 -15 -5 -6 -2 8 7 -23 10 -19 -10 0	5.598332050807308 513.142778328943998

Note

For the first test case, the corresponding image is as follows:



For the second test case, the corresponding image is as follows:


30
25
20
15
10
5
−30
−25
−20
−15
−10
−5
0
5
10
15
20
25
30
−5
−10
−15
−20
−25
−30

Problem C. Finding Keywords

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 megabytes

In search algorithms, we often encounter the following problem: there are many entries and a sequence of keywords, and you want to find entries that match the keywords. For example, the entry "2025 China Collegiate Programming Contest (CCPC) Jinan Regional Contest" matches the keywords "2025 CCPC Jinan". Note that the keyword sequence does not need to appear consecutively in the entry; it only needs to appear as a subsequence of the entry.

Based on practical considerations, we can define the following relevance algorithm: select as many **non-overlapping** subsequences from the entry as possible, such that each subsequence is exactly equal to the given keyword sequence; then the relevance is the number of subsequences.

In this problem, each word can be represented by a positive integer. Given a keyword sequence of length m, to simplify the problem, the keywords are **all distinct**. Additionally, n entries are provided, where the i-th entry is a sequence of positive integers of length l_i. Your goal is to calculate the relevance for each entry.

Input

The first line of input contains an integer m ($1 \le m \le 10$), representing the length of the keyword sequence.

The next line contains m integers $b_1, b_2, \ldots, b_m$ ($1 \le b_i \le m$), representing the keyword sequence. It is guaranteed that $b_1, b_2, \ldots, b_m$ are all distinct.

The next line contains an integer n ($1 \le n \le 100$), representing the number of entries.

The following n lines: the i-th line first reads an integer l_i ($1 \le l_i \le 10^6$), representing the length of the i-th entry. Then it reads l_i integers $a_{i,1}, a_{i,2}, \ldots, a_{i,l_i}$ ($1 \le a_{i,j} \le 10^6$), representing the content of the i-th entry.

It is guaranteed that the total length of all entries does not exceed 10^6.

Output

Output n lines, where the i-th line contains an integer representing the relevance of the i-th entry.

Example

standard input	standard output
4 4 2 1 3 3 8 4 2 1 3 4 2 3 1 9 1 4 2 4 1 2 1 3 3 12 1 1 2 3 4 2 1 1 2 1 3 3	1 2 1

Note

Below, the index sequences are used to represent subsequences.

For the first entry, you can choose $[1, 2, 3, 7]$ as a subsequence, at which point $[a_1, a_2, a_3, a_7] = [4, 2, 1, 3]$, which is exactly equal to the keyword sequence.

For the second entry, you can choose $[2, 3, 5, 9]$ and $[4, 6, 7, 8]$ as subsequences. Note that $[2, 3, 7, 9]$ and $[4, 6, 7, 9]$ cannot be chosen as selected subsequences because their positions overlap.

Problem D. Lighthouse

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 megabytes

Little Y is visiting a park. The map of this park can be represented by an undirected simple connected graph containing n nodes and m edges. To help travelers plan their routes, there are several cycles in the map; each cycle can be represented by a sequence of distinct nodes $e_1, e_2, \ldots, e_l$, where $l \geq 3$, and for all $1 \leq i \leq l$, there exists an edge in the graph connecting e_i and $e_{(i \bmod l)+1}$. Additionally, the set of edges connecting all nodes in order on the cycle is called the edge set of the cycle, and two cycles are considered different if and only if their edge sets are different. As Little Y continued the visit, he discovered a property: each edge in the graph appears in at most one cycle.

Currently, there is no lighting system in the park, so when night falls, the park will be completely dark. Fortunately, the staff is preparing to place lighthouses at some nodes on the map. The distance between two nodes on the graph is defined as the minimum number of edges that must be traversed to reach one node from another. Therefore, each lighthouse, in addition to illuminating its placement node, can also illuminate all nodes that are at a distance of no more than k from the placement node.

As a competitive programming contestant, Little Y naturally thought of a question: what is the minimum number of lighthouses that need to be placed to ensure that all nodes are illuminated?

Input

This problem contains multiple test cases. The first line of input contains an integer T ($1 \leq T \leq 10^5$), representing the number of test cases.

For each test case:

The first line contains three integers n, m, k ($2 \leq n \leq 2 \times 10^5, n-1 \leq m \leq 2 \times 10^5, 1 \leq k \leq n$), representing the number of nodes and edges on the map, as well as the maximum distance that a lighthouse can illuminate.

The next m lines each contain two integers u, v ($1 \leq u, v \leq n, u \neq v$), representing an edge on the map.

It is guaranteed that the given undirected graph is connected and has no multiple edges or self-loops, and each edge appears in at most one cycle. Additionally, the total sum of n and m across all test cases does not exceed 2×10^5.

Output

For each test case, output a single integer representing the minimum number of lighthouses that need to be placed.

Example

standard input	standard output
3 5 4 1 1 2 1 3 3 4 1 5 5 5 1 1 2 2 3 3 4 4 5 5 1 8 8 2 1 2 2 3 3 4 4 5 5 1 1 6 2 7 3 8	2 2 1

Note

For the first test case, two lighthouses can be placed at node 1 and node 4 respectively.

For the second test case, two lighthouses can be placed at node 2 and node 5 respectively.

For the third test case, one feasible solution is to place a lighthouse at node 2. It can be proven that this is the only solution that satisfies the condition of having the minimum number of lighthouses.

Problem E. Tree and Subgraph Problem

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 megabytes

Given an unrooted tree containing n nodes, where each node has a color represented by a positive integer c_i for the i-th node.

Assume that S and T are sets of nodes in the tree. An **ordered pair** (S, T) is considered valid if all the following conditions are met:

- Both S and T are non-empty sets;
- All nodes in S have the same color, and all nodes in T have the same color;
- The **subgraph** $\text{sub}(S)$ of the set of nodes S is defined as the **smallest connected subgraph** of the tree that contains the set of nodes S, and similarly for $\text{sub}(T)$. It is required that $\text{sub}(S)$ and $\text{sub}(T)$ do not intersect, meaning they have no common nodes or edges.

Determine the number of all valid ordered pairs (S, T) modulo 998244353. It should be noted that if $S \neq T$, then the ordered pair (S, T) is considered different from the ordered pair (T, S).

Input

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

For each test case:

The first line contains an integer n ($1 \leq n \leq 2 \times 10^5$), representing the number of nodes in the unrooted tree.

The second line contains n integers $c_1, c_2, \ldots, c_n$ ($1 \leq c_i \leq n$), representing the color of each node.

The next $n - 1$ lines each contain two integers u, v ($1 \leq u, v \leq n, u \neq v$), representing the edges of the tree. It is guaranteed that these $n - 1$ edges form a tree structure.

The data guarantees that the total sum of n across all test cases does not exceed 2×10^5.

Output

For each test case, output a single line containing an integer, representing the number of ordered pairs modulo 998244353.

Example

standard input	standard output
3 2 1 1 1 2 5 1 1 1 2 2 1 2 2 3 3 4 4 5 5 1 2 1 2 1 1 2 1 3 2 4 2 5	2 54 42

Problem F. Grid Filling Game

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 megabytes

You are playing a grid filling game. On the game interface, several squares are arranged in a row, and you need to fill each square with a **positive integer**. The system will then calculate the score based on the numbers you fill in.

Before each round of the game starts, the system will provide you with constraints for filling the squares. Suppose there are n squares arranged in a row for the current game, the system will provide two positive integers l_i, r_i for the i-th square, indicating that the number x_i you fill in the i-th square must satisfy $l_i \le x_i \le r_i$. After you have filled in the numbers, the system will separate the sequence of squares into several segments based on adjacent squares with different numbers. The system will then calculate the sum of the squares of the lengths of each segment, which is referred to as the **continuity** of the filling scheme.

For example, for the filling scheme $x = [1, 1, 3, 3, 5, 5, 5, 3, 3]$, the segmentation result is $\underline{1, 1}, \underline{3, 3}, \underline{5, 5, 5}, \underline{3, 3}$, and the continuity corresponds to $2^2 + 2^2 + 3^2 + 2^2 = 21$.

The goal of this game is to **minimize continuity**. You need to determine the minimum value of continuity under the optimal filling strategy.

Input

You need to answer for multiple rounds of the game. The first line contains an integer T ($1 \le T \le 10^5$), representing the number of game rounds.

For each round of the game:

The first line of input contains a positive integer n ($1 \le n \le 10^6$), representing the number of squares.

The next line contains n positive integers l_i ($1 \le l_i \le 10^9$), representing the lower bound constraints for each square.

The following line contains n positive integers r_i ($1 \le r_i \le 10^9, l_i \le r_i$), representing the upper bound constraints for each square.

It is guaranteed that the total sum of n across all games does not exceed 10^6.

Output

For each round of the game, output a single integer representing the minimum value of continuity.

Example

standard input	standard output
3 3 1 1 4 5 1 4 6 1 2 3 4 4 4 3 3 3 4 5 6 7 1 1 2 2 1 1 1 1 2 2 2 2 1 1	3 6 17

Note

For the first round of the game, one possible filling scheme is $x = [3, 1, 4]$, which corresponds to a continuity of $1^2 + 1^2 + 1^2 = 3$.

For the second round of the game, one possible filling scheme is $x = [1, 2, 3, 4, 5, 6]$, which corresponds to a continuity of $1^2 + 1^2 + 1^2 + 1^2 + 1^2 + 1^2 = 6$.

For the third round of the game, one possible filling scheme is $x = [1, 1, 2, 2, 2, 1, 1]$, which corresponds to a continuity of $2^2 + 3^2 + 2^2 = 17$.

Problem G. Make SYSU Great Again IV

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	1024 megabytes

As you know, you are participating in the Chinese Constructive Problem Contest (CCPC), and without a doubt, the proposing school, Sun Yat-sen University, as the Kingdom of Construction, is very eager for you to challenge some related problems.

Little U is a student in the Kingdom of Construction. In an algorithm class, Little U learned about the MCOP (Matrix Chain Ordering Problem). Here is an introduction to MCOP:

Given a sequence of positive integers of length $n \geq 3$, $[w_1, w_2, \ldots, w_n]$, as input to the algorithm. The following formula multiplies $n-1$ matrices to ultimately obtain a new matrix:

$$M = M_1 \times M_2 \times \cdots \times M_{n-1}$$

where M_i is a matrix of size $w_i \times w_{i+1}$. Matrix multiplication satisfies the associative law, so changing the order of multiplication will yield the same final matrix M, but the number of operations may differ. The goal of the problem is to find an optimal multiplication strategy among $(n-2)!$ possible strategies, such that the total number of operations required for matrix multiplication is minimized. Here, it is assumed that multiplying matrices of sizes $p \times q$ and $q \times r$ will yield a matrix of size $p \times r$, requiring pqr operations. The minimum total number of operations required to compute M under the optimal strategy is denoted as $\mathrm{MCOP}([w_1, w_2, \ldots, w_n])$.

Little U has learned how to solve this problem with sufficiently good time complexity in class, but as a student of the Kingdom of Construction, Little U naturally thought of the following problem:

Given an integer X, satisfying $1 \leq X \leq 10^9$. You need to construct an integer sequence of length l, $[a_1, a_2, \ldots, a_l]$, that meets the following requirements:

- $3 \leq l \leq 6$;
- For all $i \in [1, l]$, $1 \leq a_i \leq 4000$;
- $\mathrm{MCOP}([a_1, a_2, \ldots, a_l]) = X$.

Little U has already implemented a checker using the algorithm learned in class, and what you need to do is solve this construction problem.

Input

This problem contains multiple test cases. The first line of input contains an integer T ($1 \leq T \leq 500$), representing the number of test cases.

For each test case, the input consists of a single line with an integer X ($1 \leq X \leq 10^9$), the meaning of which has been given in the problem.

Output

For each test case: first output a line with an integer l ($3 \leq l \leq 6$), representing the length of the constructed sequence. Then output a line with l integers, representing the constructed sequence.

It can be proven that for any input X, there is always a construction that meets the requirements. If there are multiple constructions that meet the requirements, output any one of them.

Example

standard input	standard output
2 90 10	4 5 3 2 6 3 1 1 10

Note

For the first test case, the constructed sequence is $[5, 3, 2, 6]$, which represents the need to perform a chain multiplication of three matrices of sizes 5×3, 3×2, and 2×6. Consider two multiplication strategies:

- First, multiply M_1 and M_2 to obtain a matrix of size 5×2, then multiply with M_3, corresponding to the expression $(M_1 \times M_2) \times M_3$. The total number of operations at this point is $5 \times 3 \times 2 + 5 \times 2 \times 6 = 90$;
- First, multiply M_2 and M_3 to obtain a matrix of size 3×6, then multiply with M_1, corresponding to the expression $M_1 \times (M_2 \times M_3)$. The total number of operations at this point is $3 \times 2 \times 6 + 5 \times 3 \times 6 = 126$.

Here, the goal is to minimize the total number of operations, so $\mathrm{MCOP}([5, 3, 2, 6]) = \min(90, 126) = 90$.

Problem H. Hashing

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	1024 megabytes

To determine whether rooted trees are isomorphic (in this problem, you do not need prior knowledge of tree isomorphism), a common algorithm is to use tree hashing. With a suitable hash function, non-isomorphic rooted trees are likely to yield different hash values, but hash collisions are also difficult to avoid.

There are various implementations of hash functions, but some hash function designs are not very correct. Little C has such a "not very correct"hash function, defined as follows: Let sz_x be the number of points contained in the subtree of x, and son_x be the set of all child nodes of x. For a rooted tree with r as the root, its hash value $h(r)$ is defined as:

$$h(r) = 1 + \sum_{v \in son_r} h(v) \times f(sz_v)$$

where f represents any mapping from the set of positive integers to the set of positive integers, i.e., $f \in \{\varphi : \mathbb{Z}^+ \to \mathbb{Z}^+\}$. In particular, the hash value for a rooted tree with only one node is 1.

Next, Little C defines that for two rooted trees, if for all choices of f, it is impossible to compute different hash values, then the two trees are said to be indistinguishable; otherwise, if there exists a choice of f such that the two trees can compute different hash values, then the two trees are said to be distinguishable.

Little C wants to use this hash function to generate as many **rooted binary trees** as possible, which are rooted trees where each node has at most two child nodes. Little C hopes you can calculate the maximum number of distinguishable rooted binary trees containing n nodes. Of course, the answer may be very large, so Little C has prepared a modulus p for you, and you need to take your result modulo p.

Input

This problem contains multiple test cases. The first line of input contains an integer T ($1 \le T \le 10000$), representing the number of test cases.

For each test case, the input consists of a single line with two integers n, p ($1 \le n \le 3000, 2 \le p \le 10^9+9$), representing the number of nodes in the rooted tree and the modulus provided by Little C.

It is guaranteed that the total sum of n for all test cases does not exceed 10000.

Output

For each test case, output a single line with an integer, representing the result of the answer modulo p.

Example

standard input	standard output
8 1 1000000009 5 1000000009 10 1000000009 15 1000000009 100 1000000009 150 1000000009 1000 1000000009 1500 1000000009	1 6 207 10904 478868953 202933416 152259136 264735211

Problem I. I Love CCPC

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	1024 megabytes

A string t of length k consisting of lowercase letters is called a CCPC (Character-Character-Palindrome-Character) string if and only if it satisfies the following properties:

- The length of t is at least 4, i.e., $k \geq 4$;
- The first character, the second character, and the last character of t are equal, i.e., $t_1 = t_2 = t_k$;
- The remaining part of t is a palindrome, i.e., $t_3 t_4 ... t_{k-1}$ is a palindrome.

For example, `ccpc`, `ppap`, `zzzz`, and `aabcddcba` are all CCPC strings, while `jinan`, `aaa`, and `oovoo` are not.

A substring of a string s can be represented using two indices $1 \leq l \leq r \leq |s|$, corresponding to the content $s_l s_{l+1} \ldots s_r$. In this problem, two substrings are considered different if and only if the chosen indices l, r are not the same. Given a string s that contains only lowercase letters, first calculate the number of CCPC substrings. Then n operations will be applied, where each operation inserts a lowercase letter on the **left or right** side of s. You need to determine the number of CCPC substrings after each insertion.

Input

This problem contains multiple test cases. The first line of input contains an integer T ($1 \leq T \leq 100$), representing the number of test cases.

For each test case:

The first line contains a string s consisting only of lowercase letters ($4 \leq |s| \leq 5 \times 10^5$), representing the initial content of the string.

The next line contains an integer n ($1 \leq n \leq 5 \times 10^5$), representing the number of operations.

The following line contains n strings of length 2, $c_1, c_2, \ldots, c_n$, separated by spaces. Here, c_i indicates the content of the i-th operation: the first character is `L` indicating an insertion on the left side, and `R` indicating an insertion on the right side, while the second character represents the lowercase letter to be inserted.

It is guaranteed that the sum of $|s|$ across all test cases and the sum of n do not exceed 5×10^5.

Output

For each test case, output a line with $n + 1$ integers. The first integer represents the answer in the initial case, followed by n integers, where the i-th integer represents the answer after the first i operations.

Example

standard input	standard output
2 iloveccpc 6 Rj Ri Ln La Ln Ln cpcpccp 9 Lc Lc Rp Rc Lc Lc Rp Rp Rc	1 1 1 1 1 1 2 0 2 3 3 4 5 7 8 8 9

Note

For the second test case, after all operations are completed, the content of the string s is cccccpcpccppcppc. The CCPC substrings include: $s_{1...4}$ = cccc, $s_{1...5}$ = ccccc, $s_{2...5}$ = cccc, $s_{3...10}$ = cccpcpcc, $s_{4...7}$ = ccpc, $s_{4...9}$ = ccpcpc, $s_{9...13}$ = ccppc, $s_{9...16}$ = ccppcppc, and $s_{11...14}$ = ppcp.

Problem J. Memory, Permutation, and Rooted Tree

Input file:	`standard input`
Output file:	`standard output`
Time limit:	5 seconds
Memory limit:	1024 megabytes

One day, Little P received a tree with n nodes, rooted at node 1, and a permutation p_i of length n. Little P recalled a problem:

Given a rooted tree and a permutation. For each node u in the tree, a node v in the **subtree** of u is called important if and only if $u \neq v$ and the position of v in the permutation appears before that of u. Let d_u be the minimum distance from u to any important node. Specifically, if there are no important nodes for u, then let $d_u = -1$. The distance between two points in the tree is defined as the number of edges in the simple path between them.

After exhausting all efforts to calculate d_i for each node i, Little P set the data aside. As time passed, Little P revisited this problem, where the structure of the rooted tree and the d_i array remained, but the permutation p_i had disappeared. Now, Little P hopes that you can provide a possible permutation p_i from the available information. If there are multiple possible permutations p_i, you need to find the **lexicographically smallest** one.

For two permutations a and b of length n, we say that the lexicographic order of a is less than that of b if and only if there exists $1 \leq i \leq n$ such that $a_i < b_i$, and for all $1 \leq j < i$, $a_j = b_j$.

Input

This problem contains multiple test cases. The first line of input contains an integer T ($1 \leq T \leq 5 \times 10^4$), representing the number of test cases.

For each test case:

The first line contains an integer n ($2 \leq n \leq 5 \times 10^5$), representing the number of nodes in the rooted tree.

The second line contains n integers $d_1, d_2, \ldots, d_n$ ($-1 \leq d_i \leq n$), the meanings of which have been given in the problem.

The next $n-1$ lines each contain two integers u, v ($1 \leq u, v \leq n, u \neq v$), indicating that there is an edge connecting node u and node v. It is guaranteed that these $n-1$ edges form a tree structure.

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

Output

For each test case, if there exists at least one permutation that satisfies the requirements, output a line with n positive integers representing the lexicographically smallest permutation; otherwise, output a line with `-1`.

Example

standard input	standard output
3 5 -1 1 -1 -1 -1 1 2 2 3 2 4 1 5 10 0 1 1 1 -1 -1 -1 -1 -1 -1 5 3 4 3 8 4 4 2 4 1 2 10 9 5 5 7 6 1 10 2 -1 -1 -1 1 -1 -1 -1 -1 1 2 6 6 10 5 6 5 7 8 10 3 6 5 4 6 9 1 10	1 3 2 4 5 -1 6 1 2 3 4 5 7 8 9 10

Note

For the first test case, the permutation $p = [1, 2, 3, 4, 5]$ is not feasible because, under this permutation, there are no important nodes for node 2, thus $d_2 = -1 \neq 1$. Similarly, the permutation $p = [3, 4, 1, 5, 2]$ is also not feasible because, under this permutation, $d_1 = 2 \neq -1$.

Problem K. Card Game

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 megabytes

Alice and Bob are playing a card game. They have collected $2n$ cards, each of which is written with an integer. Also, each integer between 1 and n appears on **exactly two cards**. Initially, Alice and Bob each hold n cards.

The game will proceed in turns, with Alice going first. On each turn, the current player must choose a card from their hand and place it on top of the pile. If there are now two cards with the same number in the pile, the current player scores 1 point and removes the two matching cards along with all the cards in between, placing them in the discard pile.

Alice is a novice at this game, and Bob wants to tease her. Bob wants to know: for a given sequence of moves by Alice, how can he arrange his moves to minimize Alice's score?

Input

The input consists of multiple test cases. The first line contains an integer T ($1 \le T \le 10^5$), representing the number of test cases.

For each test case:

The first line contains an integer n ($1 \le n \le 5 \times 10^5$), representing the number of different card types.

The next line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$), representing Alice's sequence of moves. It is guaranteed that no number appears more than twice in the sequence a_i.

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

Output

For each test case:

Output an integer on the first line, representing Alice's minimum score.

On the second line, output n integers $b_1, b_2, \ldots, b_n$ ($1 \le b_i \le n$), representing one of Bob's sequences of moves. It must be ensured that each number from 1 to n appears exactly twice in the combined sequences of a_i and b_i. If there are multiple sequences that minimize Alice's score, any one of them may be output.

Example

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

Note

For the third test case, the events corresponding to each round of operations and the state of the pile after each operation are as follows:

#	**Operation**	**Pile**	**Alice's Score**
1	Alice plays a 1, no two cards with the same number in the pile	[1]	0
2	Bob plays a 3, no two cards with the same number in the pile	[1, 3]	0
3	Alice plays a 4, no two cards with the same number in the pile	[1, 3, 4]	0
4	Bob plays a 2, no two cards with the same number in the pile	[1, 3, 4, 2]	0
5	Alice plays a 5, no two cards with the same number in the pile	[1, 3, 4, 2, 5]	0
6	Bob plays a 2, two 2s in the pile, Bob scores 1 point	[1, 3, 4]	0
7	Alice plays a 1, two 1s in the pile, Alice scores 1 point	[]	1
8	Bob plays a 3, no two cards with the same number in the pile	[3]	1
9	Alice plays a 4, no two cards with the same number in the pile	[3, 4]	1
10	Bob plays a 5, no two cards with the same number in the pile	[3, 4, 5]	1

Problem L. Activity Rehearsal

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	1024 megabytes

Little J is currently rehearsing for an activity. Now, n students are lined up in order, each holding a sign with a number on it. It is assumed that the number on the sign held by the i-th student from left to right is a_i. The numbers on the signs are all distinct integers within the range of 1 to n.

Now, a "comparison"activity rehearsal is to be conducted. This activity will take a total of $n-1$ rounds. In each round, the two students at the far left will step forward, and then they will compare the numbers on their signs. The student with the larger number will be **selected**, while the student with the smaller number will return to the far right. **Note that the selected student will not return to the lineup and will not participate in subsequent rounds.**

Little J wants to know who is selected in each round under different arrangements of numbers. He will perform m operations, each of which is one of the following two types:

- Choose two positions x, y such that $1 \le x < y \le n$, and then swap the signs held by the x-th and y-th students;
- Given two parameters l, r such that $1 \le l \le r < n$, **simulate** a "comparison"activity and calculate the sum of the numbers on the signs of the selected students between the l-th round and the r-th round. After the simulation, all students will **return to their original positions**.

Input

The first line of input contains two integers n, m ($2 \le n \le 10^5, 1 \le m \le 10^5$), representing the number of students and the number of operations.

The next line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$), representing the initial numbers on the signs held by each student. It is guaranteed that all a_i are distinct.

The following m lines each contain a character and two integers, representing an operation.

- If the input format is `C` x y, it means to swap the signs held by the x-th and y-th students, with the guarantee that $1 \le x < y \le n$;
- If the input format is `A` l r, it means to simulate a "comparison"activity and calculate the sum of the selected numbers between the l-th round and the r-th round, with the guarantee that $1 \le l \le r < n$.

Output

For each simulation operation, output a single integer on a new line, representing the answer.

Example

standard input	standard output
5 6 4 3 1 2 5 A 3 4 C 1 2 A 3 4 A 2 4 C 3 5 A 1 3	8 8 10 12

Note

In the first simulation operation, the numbers held by the students in order are $[4, 3, 1, 2, 5]$. In the four rounds of comparison, the selected numbers are $4, 2, 5, 3$, so the sum of the selected numbers between the 3-rd round and the 4-th round is $5 + 3 = 8$.

In the last simulation operation, the numbers held by the students in order are $[3, 4, 5, 2, 1]$. In the four rounds of comparison, the selected numbers are $4, 5, 3, 2$, so the sum of the selected numbers between the 1-st round and the 3-rd round is $4 + 5 + 3 = 12$.

Problem M. MEX Problem

Input file:	standard input
Output file:	standard output
Time limit:	4 seconds
Memory limit:	1024 megabytes

Little N has recently learned about MEX. MEX (Minimum Excluded) refers to the smallest non-negative integer that does not appear in an array. For example: $\text{MEX}([3,1,0]) = 2, \text{MEX}([2,2,1]) = 0, \text{MEX}([0,3,1,2]) = 4$. Little N defines that for a non-negative integer array of length k, $[b_1, b_2, \ldots, b_k]$, if $\text{MEX}([b_1, b_2, \ldots, b_k]) = k$, then the array b is called a k — good array.

Given a positive integer n, for an n — good array $p_1, p_2, \ldots, p_n$, Little N defines the function $f(p)$ as the number of integer pairs (l, r) that satisfy the following conditions:

- $1 \le l \le r \le n$;
- $p_l, \ldots, p_r$ is a $(r - l + 1)$ — good array.

Little N has proven the following theorem: for any n — good array p, there holds $1 \le f(p) \le n$. Now, Little N hopes you can find the number of n — good arrays p for each $i \in [1, n]$ such that $f(p) = i$. Of course, this number may be very large, so you need to provide the result modulo 998244353.

Input

The input consists of a single line containing an integer n ($1 \le n \le 10^5$).

Output

Output a single line containing n integers, where the i-th integer represents the number of n — good arrays p satisfying $f(p) = i$, modulo 998244353.

Examples

standard input	standard output
1	1
3	0 2 4
7	0 2312 1424 728 352 160 64

Note

For $n = 3$, it can be proven that there are six 3 — good arrays, with corresponding function values:

$$f([0,1,2]) = 3, f([0,2,1]) = 2, f([1,0,2]) = 3, f([1,2,0]) = 2, f([2,0,1]) = 3, f([2,1,0]) = 3$$