The 2nd Universal Cup

Stage 25: Shenzhen

March 9-10, 2024

This problem set should contain 13 problems on 20 numbered pages.

Based on

China Collegiate Programming Contest (CCPC)

Hosted by

Prepared by

Problem A. A Good Problem

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

A good problem should have a concise statement.

You are given an array a of length n, initially filled with zeros, and another array b of length n. Your goal is to transform array a into array b. You can perform the following two types of operations:

- 1 x: Add 1 to all elements in a that are equal to x.
- 2 x: Add 1 to the element in a at index x.

You can perform no more than 20 000 operations.

Input

The first line contains a positive integer n ($1 \le n \le 1000$).

The second line contains n non-negative integers representing the array b ($0 \le b_i \le n$).

Output

The first line should contain an integer k, representing the number of operations.

The following k lines should each contain two integers 1 x or 2 x, representing an operation. For the operation type 1 x, you must ensure that $0 \le x \le n$. For the operation type 2 x, you must ensure that $1 \le x \le n$.

Example

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

Problem B. Excuse

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

You are given a fair coin (That is, if you flip this coin, there will be a 50% chance of heads and 50% chance of tails.). You would like to use it to generate a sequence of n integers $a_1, a_2, \cdots, a_n$. To do that, you will repeat the following process exactly n times:

- Keep tossing this coin until you get a tails-up result.
- Suppose you get k times of heads-up result before you stop.
- An integer k is generated and added to the end of the sequence.

For a sequence of integers $b_1, b_2, \cdots, b_m$, let $\operatorname{mex}(b_1, b_2, \cdots, b_m)$ be the smallest non-negative integers x such that:

- For each $1 \le i < x$, there exists $1 \le j \le m$ such that $b_j = i$.
- $b_j \neq x$ for all $1 \le j \le m$.

For example, $\operatorname{mex}(0, 1, 0, 3) = 2$, $\operatorname{mex}(4, 3, 2, 1, 0, 6, 7, 5) = 8$ and $\operatorname{mex}(1, 2, 3) = 0$.

Now, you would like to calculate the expected value of $\operatorname{mex}(a_1, a_2, \cdots, a_n)$, modulo 998 244 353.

Input

The first line of the input contains a single integer n ($1 \le n \le 10^5$).

Output

Output a single line contains a single integer, indicating the answer modulo 998 244 353.

Examples

standard input	standard output
1	499122177
3	561512450

Note

In the first example:

- If $a_1 = 0$, we have $\operatorname{mex}(a_1) = 1$. There is a 1/2 probability of this happening.
- Otherwise, we have $\operatorname{mex}(a_1) = 0$.

Therefore, the answer is $1 \times \frac{1}{2} + 0 \times (1 - \frac{1}{2}) = \frac{1}{2} \equiv 499122177 \pmod{998\,244\,353}$

Problem C. Count off 3

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

Xiao Z and Xiao R have started playing the counting game again. This time, facing the predicament of running out of new twists for their game and Xiao R going abroad for an exchange program, they find themselves unable to play together in person and can only communicate online.

However, the idea of playing online sparked a new inspiration. As is well-known, data is represented in binary form. Previously, they played the game in decimal, but this time they proposed playing in binary instead.

In the traditional counting game, players take turns counting numbers, and any number that is a multiple of 7 or contains the digit 7 must be skipped. Unfortunately, they quickly realized that this rule isn't very interesting in binary since the digit 7 doesn't exist in binary, and whether a number is a multiple of 7 is independent of its representation in any base.

But what if they looked at it from another angle? The same string of digits can represent vastly different numbers in different bases. For example, the string "1010" represents 10 in binary ($1010_2 = 0 \cdot 2^0 + 1 \cdot 2^1 + 0 \cdot 2^2 + 1 \cdot 2^3 = 10$), but represents 30 in ternary ($1010_3 = 0 \cdot 3^0 + 1 \cdot 3^1 + 0 \cdot 3^2 + 1 \cdot 3^3 = 30$). Following this logic, how many strings of digits are there that, in any base, are not multiples of 7? They decided to make a rule that only such numbers could be counted.

Therefore, they defined the following formal rules:

A string s is defined as a *valid 01 string* if and only if s is non-empty, consists only of the characters '0' and '1', and does not start with '0'.

They also defined a *base function* $f(s, k)$, where s is a valid 01 string and k is an integer not less than 2. If s has a length of n, indexed from 0, then $f(s, k) = \sum_{i=0}^{n-1} s_{n-i-1} \cdot k^i$, noting that s_0 is the high-order bit and s_{n-1} is the low-order bit.

Given a valid 01 string s as the *upper limit* for counting, they wanted to know how many strings (modulo $10^9 + 7$), not exceeding s, are there that are not multiples of 7 in any base? In other words, how many valid 01 strings s' are there, satisfying that for any integer $k \geq 2$, 7 does not divide $f(s', k)$, where "not exceeding s" means in the binary sense, i.e., $f(s', 2) \leq f(s, 2)$?

Input

There are multiple test cases. The first line of the input contains a single integer T ($1 \leq T \leq 10$), indicating the number of the test cases. For each of the test case:

The first line of the input contains a single valid 01 string s ($|s| \leq 10^4$), indicating the *upper limit* for counting.

Output

For each test case, output a single line contains a single integer, indicating the answer. Since the answer can be huge, you need to output it modulo $10^9 + 7$.

Example

standard input	standard output
5 1 1010 110101 1000111000 101101001000	1 2 15 114 514

Problem D. Bot Brothers

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

Doodle and Doddle are robot brothers who love playing games together.

Today's game is as follows:

There is a rooted tree with n nodes, among which there are m ($m \geq 3$) leaves (nodes with degree 1 and not the root node), and the leaves are numbered in a permutation from 1 to m.

Doodle and Doddle initially stand at the root node n, and they take turns performing the following operations, with Doodle going first:

- If the current node is a leaf, do nothing.
- If the current node is not a leaf, choose one of its child nodes in the tree and move to that child node.

When both players reach a leaf, the game ends. Let the leaf where Doodle stands be numbered x, and Doddle be numbered y.

- If $x \bmod m = (y + 1) \bmod m$, then Doodle wins.
- If $(x + 1) \bmod m = y \bmod m$, then Doddle wins.
- Otherwise, it's a tie.

Doodle and Doddle are both extremely smart robots, so they will definitely adopt the optimal strategy.

Please determine who will win in the end.

Input

There are multiple test cases. The first line of the input contains a single integer T ($1 \leq T \leq 10^5$), indicating the number of the test cases. For each of the test case:

The first line contains two integers n ($4 \leq n \leq 10^5$) and m ($3 \leq m < n$), indicating the number of nodes in the tree and the number of leaves, respectively.

The following $n - 1$ lines each contain two integers x and y ($1 \leq x, y \leq n$), indicating an edge in the tree. It is guaranteed that nodes $1, 2, \ldots, m$ are exactly the leaves of the tree, and node n is the root of the tree.

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

Output

For each test case, output a single line `Doodle` or `Doddle`, indicating the winner. If the game ties, output a single line `Tie`.

Example

standard input	standard output
2 6 3 1 4 2 4 3 5 5 6 4 6 5 4 1 5 2 5 3 5 4 5	Tie Doddle

Problem E. Two in One

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

Given a color sequence c of length n.

Define $\mathrm{occ}(x, l, r)$ as the number of occurrences of color x in the sequence c between the l^{th} and r^{th} items.

Please find an interval $[l, r]$ and then find two colors x, y (note x, y can be equal), so that $\mathrm{occ}(x, l, r)\,\mathrm{or}\,\mathrm{occ}(y, l, r)$ is maximized, where or is the binary OR operation.

Input

There are multiple test cases. The first line of the input contains a single integer T ($1 \le T \le 10^5$), indicating the number of the test cases. For each of the test case:

For each test case, the first line contains a positive integer n ($1 \le n \le 10^5$).

The second line contains n integers, representing the sequence c ($1 \le c_i \le n$).

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

Output

Output a single line contains a single integer, indicating the answer.

Examples

standard input	standard output
1 7 1 2 3 4 3 2 1	3
1 9 1 1 1 1 1 2 2 2 2	7

Note

For the first test case, one possible selection is the interval $[2, 5]$ and choosing colors 2 and 3, with occurrence counts of 2 and 1 respectively, resulting in a bitwise OR result of 3. It can be proven that there is no better solution.

Problem F. Gift

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Given an undirected graph with n vertices and n edges, you need to calculate how many ways there are to choose a vertex p and an edge (x, y) such that, after removing the edge (x, y), the graph becomes a tree, and when this tree is rooted at p, each node has no more than 3 children. It is guaranteed that there is at least one possible plan.

Input

The first line of the input contains a single integer n ($2 \le n \le 10^5$).

The next n lines of the input describes the edges of the graph. The i-th line of these lines contains two integers x_i and y_i ($1 \le x_i, y_i \le n$, $x_i \ne y_i$), indicating the i-th edge.

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

Output

Output a single line contains a single integer, indicating the answer.

Example

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

Problem G. Gene

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2.5 seconds
Memory limit:	512 megabytes

Let s and t be two strings with equal lengths M. Define $f(s,t) = \sum_{i=1}^{M} [s_i \neq t_i]$.

You are given N strings $s_1, s_2, \cdots, s_N$ and a constant threshold K. Each of the string contains exactly M lowercase letters. You need to perform the following queries Q times:

- Given a string t of length M, calculate $\sum_{i=1}^{N} [f(s_i,t) \leq K]$

Input

The first line of the input contains four integers N, Q, M, and K ($1 \leq N, Q \leq 300$, $1 \leq M \leq 60,000$, $1 \leq K \leq 10$).

The i-th line of the next N lines contains a single string s_i consisting exactly M lowercase letters.

The i-th line of the next Q lines contains a single string t consisting exactly M lowercase letters, indicating a query.

Output

For each query, output a single line contains a single integer, indicating the answer.

Examples

standard input	standard output
6 4 4 1 kaki kika manu nana tepu tero kaka mana teri anan	2 2 1 0
8 6 7 3 delphis aduncus peronii plumbea clymene hectori griseus electra delphis helpiii perphii clumeee eleelea ddlpcus	1 1 2 2 1 2

Problem H. Fast Hash Transform

Input file:	`standard input`
Output file:	`standard output`
Time limit:	5 seconds
Memory limit:	512 megabytes

Define a closed function f on $\mathcal{V} = \{0, 1 \cdots, 2^{64} - 1\}$ as a single-round hash function if and only if it has the following standard form:

$$f(X) = B \oplus \bigoplus_{j=1}^{m} \left((X \lll s_j) \circ_j A_j \right),$$

where,

- $A_j, B \in \mathcal{V}$;
- $0 \le m \le 64$, $0 \le s_j \le 63$ and all s_j are distinct;
  - $\oplus$ denotes bitwise XOR operation, $\lll$ denotes circular left shift operation (the circular left shift on $\mathcal{V}$ is defined as circular left shifting the unsigned binary representation padded to 64 bits), and $\circ_j$ denotes either the bitwise OR operation (denoted as $\vee$) or the bitwise AND operation (denoted as $\wedge$).

Given N single-round hash functions $f_1, f_2, \cdots, f_N$ and Q operations, the operations are of the following two types:

- Given l, r, x, compute the value of $f_r\left(\cdots\left(f_{l+1}\left(f_l(x)\right)\right)\cdots\right)$;
- Given l, modify f_l to a new single-round hash function.

Considering the practical aspect that modifying single-round hash functions would invalidate a large number of quick hash transformation results, it is guaranteed that such modifications will not exceed C times.

Input

The first line contains three non-negative integers N, Q, C, representing the number of single-round hash functions to be maintained, the number of operations, and the number of operations to modify a single-round hash function, respectively. It is guaranteed that $1 \le N, Q \le 20,000$, $0 \le C \le 400$, and $C < Q$.

From the second line to the $(N + 1)$-th line, the $(i + 1)$-th line describes the initial parameters of f_i:

- The first non-negative integer m in each line indicates the number of terms involved in the XOR sum in the standard form of f_i. It is guaranteed that $0 \le m \le 64$.
- Next, input $3m$ non-negative integers in sequence, where the $(3j - 2)$-th, $(3j - 1)$-th, and $3j$-th integers are s_j, o_j, A_j, respectively, representing the number of bits X is circularly left-shifted in the j-th term of the XOR sum, the type of bitwise operation $\circ_j$ in the j-th term ($o_j = 0$ corresponds to bitwise OR operation, $o_j = 1$ corresponds to bitwise AND operation), and the constant term of the bitwise operation in the j-th term. It is guaranteed that $0 \le s_j \le 63$ and the s_j values are distinct within the same single-round hash function, $0 \le o_j \le 1$, $0 \le A_j < 2^{64}$.
- The last number in each line is a non-negative integer B, representing the constant term in the outer layer of the standard form. It is guaranteed that $0 \le B < 2^{64}$.

From the $(N+1)$-th to the $(N+Q+1)$-th line, the $(N+i+1)$-th line describes the i-th operation.

- The beginning of each line is a non-negative integer op, indicating the type of operation.
- If $op = 0$, it indicates an operation to compute a quick hash transformation. It is followed by three non-negative integers l, r, x, indicating the starting position, ending position, and initial value of the quick hash transformation, respectively. It is guaranteed that $1 \le l \le r \le N$, $0 \le x < 2^{64}$.
- If $op = 1$, it indicates an operation to modify a certain single-round hash function. First, input an integer l, indicating the number of the single-round hash function to be modified. Then, input several integers in sequence to describe the modified f_l. The format for describing the new single-round hash function is the same as the format for describing the initial parameters of each function. It is guaranteed that $1 \le l \le N$.

Output

For each of the operation of the first type, output a single line contains a single integer, indicating the answer.

Example

standard input	standard output
3 5 1 1 4 0 0 51966 1 60 0 0 0 1 0 0 16 15 0 1 1 771 0 2 2 32368 0 3 3 0 1 2 2 0 0 15 61 1 4095 46681 0 1 3 2023	64206 2023 31 1112

Note

For the first example:

The initial parameters for the 3 single-round hash functions are:

- $f_1(X) = (X \lll 4) \oplus 51996$ (where 51996 is represented in hexadecimal as '0xCAFE');
- $f_2(X) = X \lll 60$;
- $f_3(X) = (X \vee 16) \oplus 15$ (where 16 and 15 are represented in hexadecimal as '0x0010' and '0x000F' respectively).

The first operation is to compute a quick hash transformation. The result of performing a quick hash transformation on 771 ('0x0303') is $f_1(771) = (771 \lll 4) \oplus 51996 = 64206$ ('0xFACE').

The second operation is to compute a quick hash transformation. The result of performing a quick hash transformation on 32368 ('0x7E70') is $f_2(32368) = 32368 \lll 60 = 2023$ ('0x07E7').

The third operation is to compute a quick hash transformation. The result of performing a quick hash transformation on 0 ('0x0000') is $f_3(0) = (0 \vee 16) \oplus 15 = 31$ ('0x001F').

The fourth operation is to modify f_2. After modification, $f_2(X) = (X \vee 15) \oplus ((X \lll 61) \wedge 4095) \oplus 46681$ (where 4095 and 46681 are represented in hexadecimal as '0x0FFF' and '0xB659' respectively).

The fifth operation is to compute a quick hash transformation. For the initial value 2023 (‘0x07E7‘), since $f_1(2023) = 46222$ (‘0xB48E‘), $f_2(46222) = 1095$ (‘0x0447‘), and $f_3(1095) = 1112$ (‘0x0458‘), the result of the quick hash transformation is 1112.

Please note the impact of input and output efficiency on program running time.

In the example explanation, all hexadecimal representations are padded to 4 hexadecimal digits, but this does **not** mean that the input numbers do not exceed 65535.

Problem I. Indeterminate Equation

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Given positive integers n, k, find the number of positive integer solutions to the indeterminate equation $a^k - b^k = n$.

Input

The first line of input is an integer T ($1 \le T \le 20$) indicating the number of queries. The following T lines, each contain two integers n, k indicating a single query. It is guaranteed that $1 \le n \le 10^{18}$, $3 \le k \le 64$.

Output

For each query, output a single line contains a single integer, indicating the answer.

Example

standard input	standard output
3 7 3 15 4 31 5	1 1 1

Problem J. Counter Reset Problem

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

Mechanical counters are a common auxiliary tool. A classic counter typically includes a display area indicating the current count, a counting button, and a reset button. In a traditional N-digit decimal mechanical counter, a complex internal mechanical structure ensures that pressing the counting button exactly increases the original count by 1 modulo 10^N; rotating a reset knob on the side drives some digits to rotate by 1 position, and rotating the reset knob enough times can reset any initial state back to all zeros.

The internal contacts of the counter might be poor, affecting the specific behavior of the count change when rotating the reset knob. In this problem, assume that for an ideal counter, the count change pattern when rotating the reset knob once is as follows:

- (Considering leading zeros, the same below) The highest digit will always rotate by 1 position, meaning this digit increases by 1 modulo 10;
- If the highest digit and the i-th digit from the top ($2 \le i \le N$) are both d, and the digits from the second to the $(i-1)$-th positions are not greater than d, then the i-th digit rotates by 1 position, the same as the highest digit;
- Other digits not meeting the above conditions remain unchanged.

For example, when the count is "1151", rotating the reset knob once will change the count to "2251"; when the count is "9791", rotating the reset knob once will change the count to "0701".

For a given integer X (possibly with leading zeros), the minimum number of rotations of the reset knob required to rotate the counter from the initial count of X to an all-zero state is defined as the reset rotation count of X. For an N-digit decimal mechanical counter, calculate the sum of the reset rotation counts for all $X \in [L, R]$.

Input

The first line of the input contains a single integer N ($1 \le N \le 5\,000$).

The next line of the input contains two integers L and R (with exactly N digits; possibly with leading zeros; $0 \le L \le R < 10^N$).

Output

Output the sum of the reset rotation counts for all integers in $[L, R]$. Since the total sum may be large, please output a non-negative integer representing the result of the sum modulo $1,000,000,009$.

Examples

standard input	standard output
2 19 23	51
6 100084 518118	9159739
12 040139021316 234700825190	771011551

Problem K. Quad Kingdoms Chess

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

Quad Kingdoms Chess is an interesting board game. Here, we consider a simplified version, which involves two players, Player A and Player B, with the chess pieces being integers within $[1, 10^5]$. The game process is as follows:

- Initially, Player A has n_1 chess pieces, forming sequence A; Player B has n_2 chess pieces, forming sequence B. Both A and B are given, and neither Player A nor Player B can modify them.
- Both players send out the first chess piece from their sequences, assume they are x, y respectively, then a battle occurs.
  - If $x > y$, meaning Player A's piece is larger, then Player B's piece goes into a discard pile and can no longer be used. Afterwards, Player B sends out the next chess piece from the sequence to continue the battle;
  - If $y > x$, meaning Player B's piece is larger, then Player A's piece goes into a discard pile and can no longer be used. Afterwards, Player A sends out the next chess piece from the sequence to continue the battle;
  - If $x = y$, meaning both pieces are of the same size, both pieces go into the discard pile and can no longer be used. Both players then send out the next chess piece from their sequences to continue the battle.
- During the battle, if a player needs to send out a chess piece but has no remaining pieces, then that player loses and the other player wins.
  - Specifically, if both players need to send out a chess piece but neither has any remaining pieces, then the game is considered a draw.

You can refer to the sample explanation to understand the game process.

Player A and Player B are good friends, and they do not wish to have a winner in the game; a draw would make them happier.

Given the initial game situation and m modifications, in each modification, a chess piece in one of the sequences is replaced with another piece, and the effect of each modification is retained after that modification.

After each modification, they want you to help them analyze whether the game result would be a draw if played according to the sequence.

Input

The first line is a positive integer n_1 ($1 \le n_1 \le 10^5$) indicating the number of chess pieces Player A has.

The second line contains n_1 positive integers $a_1, a_2, \ldots, a_{n_1}$ ($1 \le a_i \le 10^5$) describing sequence A.

The third line is a positive integer n_2 ($1 \le n_2 \le 10^5$) indicating the number of chess pieces Player B has.

The fourth line contains n_2 positive integers $b_1, b_2, \ldots, b_{n_2}$ ($1 \le b_i \le 10^5$) describing sequence B.

The fifth line is a positive integer m ($1 \le m \le 2 \times 10^5$) describing the number of modifications.

After that, m lines, each with three positive integers o, x, y ($1 \le o \le 2$, $1 \le x \le n_o$, $1 \le y \le 10^5$), with the following meanings:

- If $o = 1$, it means to change a_x to y;

- If $o = 2$, it means to change b_x to y.

Output

For each modification, output one line. If the game result is a draw, output YES; otherwise, output NO.

Example

standard input	standard output
5 1 2 3 4 5 5 5 4 3 2 1 8 1 1 5 1 4 2 1 2 4 1 5 1 1 5 5 2 1 4 2 3 5 2 5 5	NO NO NO YES NO NO NO YES

Problem L. Rooted Tree

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

You are given a rooted tree that initially contains only a single vertex (which is the root vertex). A vertex is considered a *leaf* if it does not have any children. You have the ability to perform a specific operation exactly K times on this tree, according to the following steps:

- Choose a leaf vertex u randomly with uniform probability.
- Add M new leaf vertices as children of vertex u.

Define the depth of the vertex u (denoted by $d(u)$) as follows:

- For the root vertex, $d(u) = 0$.
- For any other vertex, $d(u) = d(v) + 1$, where v is the parent of vertex u.

Your task is to determine the expected value of the sum of the depths of all vertices in the tree after performing the specified operation K times, modulo $(10^9 + 9)$.

Input

The first line of the input contains two integers M and K ($1 \le M \le 100$, $1 \le K \le 10^7$).

Output

Output a single line contains a single integer, indicating the answer modulo $(10^9 + 9)$.

Formally, assuming the answer is simplified to the form p/q (i.e., p and q are coprime), please output x such that $qx \equiv p \pmod{(10^9 + 9)}$, and $0 \le x < 10^9 + 9$. It can be proven that x exists and is unique.

Examples

standard input	standard output
6 2	18
2 6	600000038
83 613210	424200026

Problem M. 3 Sum

Input file:	`standard input`
Output file:	`standard output`
Time limit:	0.5 seconds
Memory limit:	512 megabytes

Given n integers $a_1, \ldots, a_n$ and a modulus $M = 10^K - 1$. Find all tuples (i, j, k) $(1 \le i \le j \le k \le N)$, such that $a_i + a_j + a_k \equiv 0 \mod M$.

Input

The first line of the input contains two integers n and K $(1 \le n \le 500, 1 \le K \le 2 \times 10^4)$.

The i-th line of the next n lines contains a single integer a_i. It is guaranteed that $0 \le a_i < 10^{20\,000}$.

Output

Output a single line contains single integer, indicating the number of the tuples.

Example

standard input	standard output
4 1 0 1 10 17	3