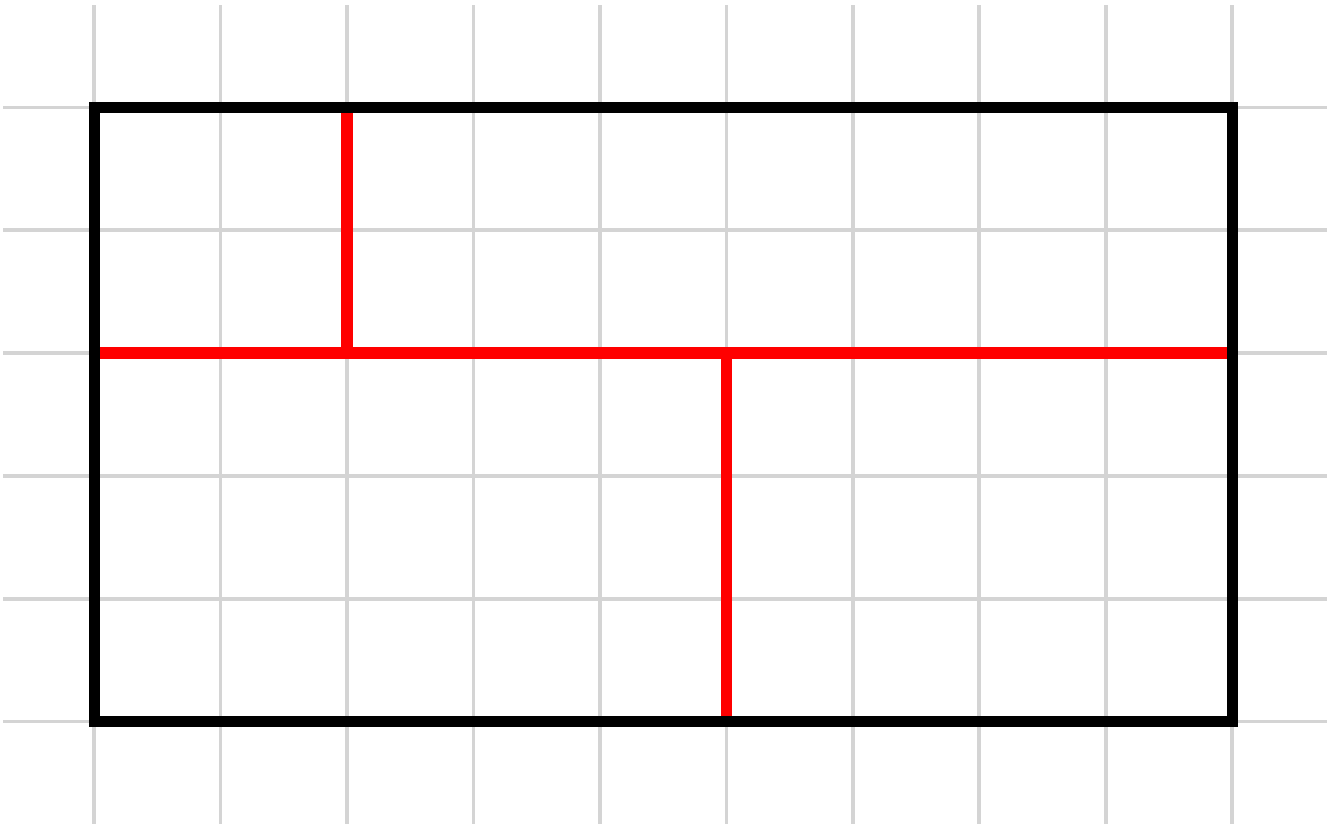
Problem A. Floor Tiles in a Park

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

Grammy is enjoying her holiday in Pigeland City Park. She was interested in the floor tiles in the park. After careful examination, she found out that each of the floor tiles is a $W \times H$ rectangle grid with vertical and/or horizontal colored segments on it. The colored segments have ends on grid points, and they split the rectangle into exactly k subrectangles.

For instance, the following illustration shows a floor tile with $W = 9$, $H = 5$, $k = 4$.



Grammy wants to know the number of different floor tiles satisfying the condition. Please tell her the answer. Since the answer may be very large, you should output the number modulo 998 244 353.

Note that two floor tiles are considered different if and only if a grid line is colored in one tile but not in the other. If two tiles can turn into the same by rotation or reflection, they may still be considered as different tiles.

Input

The only line contains three integers W, H, k ($1 \le W, H \le 10^9$, $1 \le k \le \min(5, W \cdot H)$).

Output

Output a single integer denoting the number of different floor tiles modulo 998 244 353.

Examples

standard input	standard output
2 3 5	7
4 3 5	307
6 372065168 5	114514

Problem B. Rotate Sum 3

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

Grammy loves geometry. Today, she takes out her precious convex polygon and plays with it.

Grammy thinks that symmetry is a fun characteristic for a convex polygon, so she draws out all the axes of symmetry on the polygon.

NIO is a naughty boy. He repeats the following operation several times. In each operation, he chooses an axis of symmetry as the rotation axis and rotates the polygon in three-dimensional space along the axis arbitrarily. Note that after rotating the polygon, the axes of symmetry will also rotate with the polygon.

Grammy wants to know the total volume that can be sweeped by the convex polygon during NIO's operations. Please help her.

Input

The first line contains an integer n ($3 \le n \le 10^5$), denoting the number of vertices of the convex polygon.

In each of the next n lines contains two integers x_i, y_i ($-10^9 \le x_i, y_i \le 10^9$), denoting the coordinates of the i-th vertex. The vertices are given in counterclockwise order. No three vertices lie on the same line.

Output

Output a real number, denoting the volume of the sweeped area. Your answer will be considered correct if the absolute or relative error is less than 10^{-6}.

Examples

standard input	standard output
3 0 -1 1 0 0 1	1.047197551197
3 1 1 4 5 1 4	0

Problem C. Oscar's Round Must Have a Constructive Problem

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

Grammy has a sequence A of length n.

Please find a permutation P such that $P_i \neq A_i$ for all i.

Input

There are multiple test cases.

The first line contains a single integer T ($1 \leq T \leq 100\,000$), denoting the number of test cases.

For each test case:

The first line contains a single integer n ($1 \leq n \leq 100\,000$).

The second line contains n integers $A_1, A_2, \ldots, A_n$ ($1 \leq A_i \leq n$).

It is guaranteed that the sum of n does not exceed $500\,000$.

Output

For each test case:

If the permutation does not exist, output "NO" on a single line.

Otherwise, output "YES" one the first line, then output n integers one the second line, denoting the permutation $P_1, P_2, \ldots, P_n$.

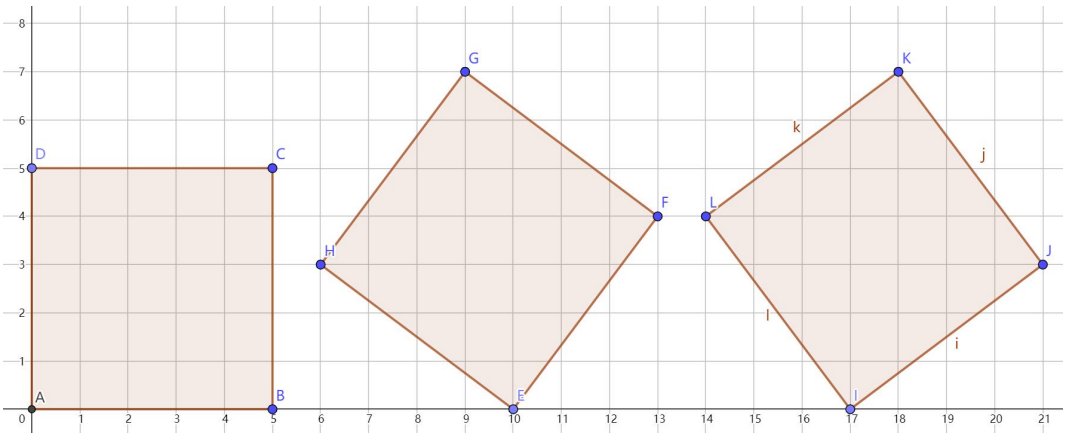
Example

standard input	standard output
3 3 3 3 3 3 3 2 1 6 1 1 4 5 1 4	NO YES 1 3 2 YES 4 5 1 2 3 6

Problem D. The Pool

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	1024 mebibytes

Marisa wants to build an $n \times m$ rectangular swimming pool for Alice. To do this, Marisa can select four integer points on an infinite two-dimensional grid, and cast magic. For example, the following picture shows three possible ways to build a 5×5 swimming pool.



Marisa soon learns that there are many ways to build the pool since four sides of the pool can be non-parallel to coordinate axes. Here two ways are considered different if and only if the pool in one way can't be translated (moved without rotation and flipping) to the pool in the other way. Now Marisa becomes curious about the total number of 1×1 squares completely inside the pool for all possible ways. As the result can be very large, you should print it modulo 998 244 353.

Input

The first line contains one integer T ($1 \le T \le 10^4$) denoting the number of test cases.

Each test case is given on a single line containing two integers n and m ($1 \le n, m \le 10^{18}$) denoting the size of swimming pool.

It is guaranteed that there are at most 10 cases where $\max(n, m) > 10^9$.

Output

For each test case, print one number, denoting the total number of 1×1 squares completely inside the pool for all possible ways (modulo 998 244 353).

Example

standard input	standard output
5 5 5 2 3 5 10 2197525579 1145141 91 65	51 12 228 438744975 34722

Note

As shown in the picture, there are exactly three different ways to build the pool. The corresponding numbers of 1×1 squares completely inside the pool in these three ways are 25, 13, and 13. So the total number is 51.

Problem E. Ternary Search

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

Recently, Grammy has learned ternary search in Tony's class. She can find the peak value in an array using this algorithm when the array is unimodal. Here, we say that an array $a_1, a_2, \ldots, a_n$ is *unimodal* if and only if it satisfies one of the following conditions:

- There exists an index k $(1 \le k \le n)$ such that $a_1 < a_2 < \ldots < a_k > a_{k+1} > \ldots > a_n$.
- There exists an index k $(1 \le k \le n)$ such that $a_1 > a_2 > \ldots > a_k < a_{k+1} < \ldots < a_n$.

As the tutor of Grammy, Tony wants to examine whether Grammy fully understands what he taught in class, so he leaves n tasks for Grammy to try ternary search. The tasks are as follows.

Initially, there is an empty array. Each task appends a **distinct** number at the right end of the array, and Grammy should do ternary search on it. However, due to Tony's carelessness, the array may not be unimodal after some addition. Since Tony has already gone to sleep, Grammy has to solve the problem by herself.

For each task, before Grammy tries ternary search on it, some operations should be performed to make it unimodal. In each operation, Grammy can swap the values of a_i and a_{i+1} for some i $(1 \le i < n)$. Grammy is a lazy girl, and she thinks that if she has to perform too many operations, she would instead wait for Tony to wake up and solve the problem. For each task, she wonders what is the least possible number of operations she has to perform to make the array unimodal. Can you help her?

Input

The input contains only a single case.

The first line contains a single integer n $(1 \le n \le 200\,000)$, denoting the number of tasks. The i-th line of the following n lines contains one integer a_i $(1 \le a_i \le 1\,000\,000\,000)$, denoting the number appended in the i-th task.

It is guaranteed that a_i are pairwise distinct.

Output

The output contains n lines. Each line contains one integer, denoting the answer to the i-th task.

Example

standard input	standard output
9 11 4 5 14 1 9 19 8 10	0 0 0 0 2 3 3 6 7

Problem F. Candies

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

Grammy has a circular array $a_1, a_2, \ldots, a_n$. You can do the following operations several (possibly zero) times in any order:

- Choose two adjacent positions with the same number, and erase them.
- Choose two adjacent positions such that the numbers on these positions add up to a special number x, and erase them.

After each time you do an operation successfully, Grammy will give you a candy. Meanwhile, the remaining parts of the array will be concatenated. For example, after deleting the third and fourth element of the array, the second element and the fifth element will become adjacent.

Find the maximum number of candies you can get.

Two positions u and v ($u < v$) are *adjacent* if and only if $u + 1 = v$ or $u = 1$ and $v = L$, where L is the length of the remaining array.

Input

The first line contains two integers n and x ($1 \le n \le 10^5$, $1 \le x \le 10^9$) denoting the length of the array and the special number x.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$) denoting the numbers in the circular array.

Output

Output an integer denoting the maximum number of candies you can get.

Examples

standard input	standard output
6 5 1 1 4 5 1 4	2
10 5 1 2 5 2 1 2 3 4 8 4	3

Problem G. Regular Expression

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

Grammy has recently been interested in regular expressions while focusing on cases where the alphabet consists of characters from 'a' to 'z'. Today she asks NIO some questions. Each question gives string A, asking the minimum length of an expression matching string A according to the matching rules, and also the number of such shortest expressions.

To learn detailed rules about how regular expressions match strings, you can refer to `https://en.wikipedia.org/wiki/Regular\_expression`.

Here we only consider characters from 'a' to 'z' and special characters '.', '?', '*', '+', '|', '(', ')'. It is assumed that the asterisk, the question mark and the plus sign have the highest priority, then follows concatenation and then alternation. Parentheses can be used to change the priority. For example, "`a(b|c)`" can match "ab" and "ac". Parentheses may be omitted when they don't change the priority. For example, "`(ab)c`" can be written as "`abc`", and "`a|(b(c*))`" can be written as "`a|bc*`".

Here are some examples of matching:

- (or): "`gray|grey`" can match "gray" or "grey".
- (question mark): "`colou?r`" matches both "color" and "colour".
- (asterisk): "`ab*c`" matches "ac", "abc", "abbc", "abbbc", and so on.
- (plus sign): "`ab+c`" matches "abc", "abbc", "abbbc", and so on, but not "ac".
- (wildcard): "`a.b`" matches any string that contains an "a", then any single character, and then "b"; and "`a.*b`" matches any string that contains an "a", and then the character "b" at some later point. More precisely, "ab" can be matched by "`a.*b`" but not by "`a.b`".
- (concatenation): Consider expression R = "`(ab|c)`" matching {"ab", "c"}, and expression S = "`(d|ef)`" matching {"d", "ef"}. Then, (RS) = "`((ab|c)(d|ef))`" matches {"abd", "abef", "cd", "cef"}.

Input

The input contains only a single case.

The first line contains a single integer Q ($1 \le Q \le 100\,000$) denoting the number of questions. The i-th line of the following Q lines contains one string A consisting of lowercase English letters ($1 \le |A| \le 200\,000$) denoting the string A of the i-th question. It is guaranteed that $\sum |A| \le 1\,000\,000$.

Output

For each question, output a single line containing two integers: the minimum length of a matching expression and the number of matching expressions of such length. Note that the answers may be extremely large, so please print them modulo 998 244 353.

Example

standard input	standard output
2 a ab	1 2 2 6

Problem H. Grammy Sorting

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

Grammy has a connected undirected graph G with n vertices numbered $1, 2, \ldots, n$. Among them are two special vertices A and B. Each vertex i has a number p_i written on it, where $p_1, p_2, \ldots, p_n$ is a permutation of $1, 2, \ldots, n$.

Grammy thinks these numbers on vertices are too chaotic. She wants to reorder the numbers such that, for each vertex x, there exists a path satisfying the following conditions:

- The path starts from A and ends at B.
- The path contains vertex x.
- The numbers along the path are strictly increasing.

Sadly, in each operation, Grammy can only choose a **simple** path starting from A and ending at an arbitrary vertex, then shift the numbers on the simple path one position nearer to the start, and put the first number to the last position. Formally, if the vertices on the simple path chosen by Grammy contain numbers $a_1, a_2, \ldots, a_{k-1}, a_k$, from start to end, then after Grammy's operation, these vertices will contain $a_2, a_3, \ldots, a_k, a_1$.

Additionally, Grammy can only operate no more than 10 000 times.

Grammy is out of ideas on how to solve this problem, so she asked you for help.

Please help Grammy to determine whether she can reorder the numbers as required. You also need to output a solution if it exists.

Input

The first line contains four integers n, m, A, B ($2 \le n \le 1000$, $1 \le m \le 2000$, $1 \le A, B \le n$, $A \ne B$).

The second line contains n integers $p_1, p_2, \ldots, p_n$ ($1 \le p_i \le n$). It is guaranteed that $p_1, p_2, \ldots, p_n$ is a permutation.

In each of the next m lines, there are two integers u_i and v_i ($1 \le u_i, v_i \le n$, $u_i \ne v_i$), denoting that there is a bidirectional edge between u_i and v_i. It is guaranteed that the graph is connected and that there is at most one edge between any two pair of vertices.

Output

If Grammy cannot properly reorder the numbers, output "`-1`" (without quotes).

Otherwise output an integer op ($0 \le op \le 10\,000$) on the first line, indicating the number of operations to perform.

On each of the following op lines, first output an integer k denoting the number of vertices on the chosen simple path. Then output k integers $x_1, x_2, \ldots, x_k$ ($x_1 = A$, $1 \le x_i \le n$), indicating the vertices on the simple path. These x_i should be distinct and form a path in G.

It can be shown that, if graph G can be properly reordered, there exists a solution with no more than 10 000 operations.

Note that you don't have to minimize op. If there are multiple solutions, output any one of them.

Examples

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

Problem I. Suffix Sort

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

Grammy has a string S of length n that consists of lowercase English letters.

For a string P, reading it left to right, write down the letters that never occurred before as $t_1, t_2, t_3, \ldots, t_k$. For example, if $P =$ "`sesame`", we write down '`s`', '`e`', '`a`', '`m`'. The *minimal representation* $R(P)$ can be obtained by replacing every occurrence of t_1 in P by the first character of the character set ("`a`"), replacing every occurrence of t_2 in P by the second character of the character set ("`b`"), and so on.

For example, when the character set is lowercase English letters, the minimal representation of "`sesame`" is "`abacdb`", the minimal representation of "`edcca`" is "`abccd`", and minimal representations R("`xy`") and R("`zt`") are both "`ab`".

Your task is to sort all suffixes of S by their minimal representation. Formally, denote suffix $S_i S_{i+1} \ldots S_{n-1} S_n$ as $S[i:]$. For two suffixes $S[i:]$ and $S[j:]$, if $R(S[i:])$ is less than $R(S[j:])$ in lexicographical order, then $S[i:]$ has to occur before $S[j:]$ in the desired order.

Please output the result as an array of indices sa: the i-th element of $sa[i]$ must be the position of the first character in the i-th smallest suffix of S in the desired order. Formally, the array must satisfy

$$R(S[sa[1]:]) < R(S[sa[2]:]) < \ldots < R(S[sa[n]:]).$$

Input

The first line contains one integers n ($1 \le n \le 200\,000$).

The next line contains a string S of length n. It is guaranteed that S only consists of lowercase English alphabets.

Output

Output n integers representing the answer.

Example

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

Problem J. Melborp Lacissalc

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

Grammy has a favorite number k. She thinks that all the numbers divisible by k are good.

For each array containing only numbers from 0 to $k-1$, Grammy defines its *goodness* as the number of non-empty consecutive subarrays that sum to a good number.

Please count the number of arrays of length n such that their goodness is t. Since the answer can be enormous, output the answer modulo 998 244 353.

Input

A single line contains three integers n, k, t ($1 \le n, k \le 64$, $0 \le t \le \frac{n(n+1)}{2}$).

Output

Output a single integer denoting the answer modulo 998 244 353.

Examples

standard input	standard output
2 5 1	12
7 10 15	2016
46 50 171	645560469

Problem K. Great Party

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

Grammy joined a great party.

There is an interesting game at the party. There are n piles of stones on the table. The i-th pile has a_i stones in it. Two players participate in the game and operate the stones in turn.

In each player's turn, the player will do the following two steps:

1. Select a **non-empty** pile of stones, select a positive amount of stones to remove from it.
2. Keep the remaining stones in the pile still **or** merge them all into another **non-empty** pile of stones.

Those who cannot operate lose the game.

Now, Grammy has q questions. For each question, she asks you how many sub-segments of $[l, r]$ satisfy that if the piles in the segment are taken out alone for the game, the first player will win.

Input

The first line contains two integers n and q ($1 \le n, q \le 10^5$).

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^6$).

The i-th of the next q lines contains two integers l_i and r_i ($1 \le l_i \le r_i \le n$).

Output

The output contains q lines. Each line contains a single integer, denoting the answer to the question.

Examples

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

Problem L. Maximum Range

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

Grammy has a simple connected undirected graph. Each of the edges has a value written on it. Please choose a simple cycle for her such that the values written on the cycle have the maximum possible range.

The *range* of a cycle is the difference between the maximum value and the minimum value written on it.

A cycle $i_1 - e_1 - i_2 - e_2 - \cdots - i_k - e_k - i_1$ (e_j is some edge connecting vertices i_j and $i_{j \bmod k+1}$ in the graph) is simple if and only if each **edge** appears at most once in it.

To prove that you really found the cycle, you need to output the vertices on the cycle in order.

It is guaranteed that there is at least one cycle in the graph.

Input

The first line contains two integers n and m ($3 \le n \le m \le 10^5$) denoting the number of vertices and the number of edges in the graph.

In each of the next m lines, there are three integers u, v, w ($1 \le u, v \le n$, $-10^9 \le w \le 10^9$, $u \ne v$), indicating that there is an edge between vertex u and vertex v having value w written on it. It is guaranteed that the graph is connected, and there is at most one edge between each pair of vertices.

Output

On the first line, output a single integer denoting the maximum range of a simple cycle in the graph.

On the second line, output a single integer k denoting the number of edges in the cycle. It is not hard to find out that the number of edges is equal to the number of vertices in the cycle.

On the last line, output k integers, denoting the vertices on the cycle in order. Note that these vertices can be repeated since only edges cannot be visited multiple times.

If there are multiple solutions, output any one of them.

Example

standard input	standard output
5 7 1 2 1 1 3 -2 2 3 1 3 4 3 4 5 1 1 5 -1 2 5 2	5 5 1 2 5 4 3

Note

In the first sample, the cycle 1-2-5-4-3-1 has the maximum range of 5, since the maximum value on the cycle is 3, and the minimum value on the cycle is -2, so the maximum range of a cycle is $3 - (-2) = 5$. It can be shown that there are no cycles with a range larger than 5.