SYSU_Collegiate_Programming_Contest
(year = 2024, status = Final)

Sun Yat-sen University

December 22, 2024

Do not open before the start of the contest.

Problem A. Anniversary Celebration

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

This year is the 100th anniversary of the founding of Sun Yat-sen University. To celebrate this great moment, the ACMM Club plans to use balloons for decoration.

The ACMM Club currently has x balloons with the letter `S`, y balloons with the letter `Y`, and z balloons with the letter `U`. The staff can bundle balloons with the four letters of `SYSU` into a set and use them for decoration. What is the maximum number of sets bundled using the available balloons?

Input

The first line contains a single integer T ($1 \le T \le 100$) — the number of test cases.

Each test case contains three integers x, y, and z ($0 \le x, y, z \le 10^4$).

Output

For each test case, output a single integer in a line — the maximum number of sets that can be bundled.

Example

standard input	standard output
3 10 2 4 11 45 14 1924 100 2024	2 5 100

Note

In the first test case, after bundling 2 sets of balloons, no balloon with the letter `Y` remains, and no new bundle can be created.

In the second test case, after bundling 5 sets of balloons, only one balloon with the letter `S` remains, but a new bundle requires two balloons with the letter `S`.

Problem B. Bruhcaea Simulator

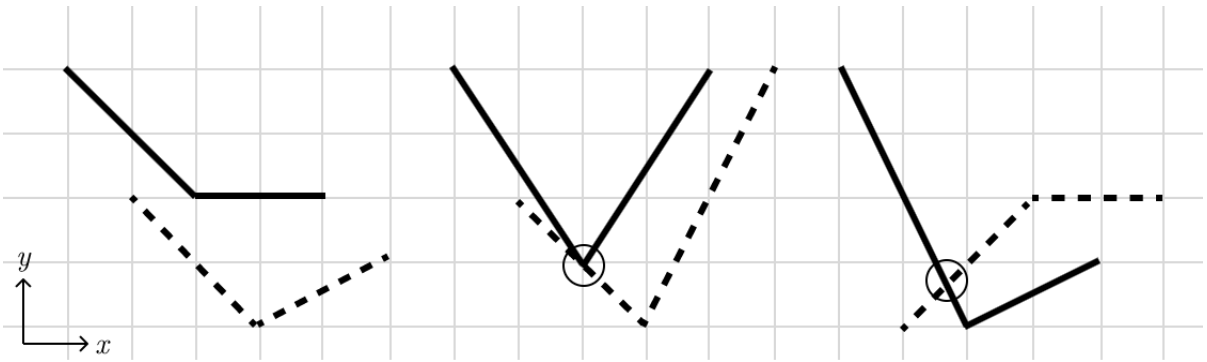
Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	512 megabytes

As I always say,
if you don't like it, don't play.

— Guy, ©lowiro

A programmer named Baker is developing a simulator for an innovative 2D rhythm game called **Bruhcaea**. There are n lines displayed during the gameplay of **Bruhcaea**, and any endpoint on the game chart can be represented by an integral point (x, y) in the Cartesian coordinate system. Here, x represents time, and y represents the line this position is on, where $1 \le y \le n$ must be satisfied for all endpoints.

There are two arcs in a chart of **Bruhcaea**, each represented as a broken line connecting m endpoints. The endpoints of the first arc are specified as $(1, p_1), (3, p_2), \ldots, (2m-1, p_m)$ in order, while the endpoints of the second arc are specified as $(2, q_1), (4, q_2), \ldots, (2m, q_m)$ in order.



The image above shows three different cases of the arcs where $n = 5$ and $m = 3$. Only the first of them satisfies the following constraint.

To maintain gameplay quality, the two arcs cannot intersect or touch **at any position** (not just at the endpoints, as shown in the third case above). Furthermore, the more movements the arcs perform, the more playful the chart becomes. Baker defines the **playfulness** of the two arcs as

$$\sum_{i=1}^{m-1} |p_i - p_{i+1}| + |q_i - q_{i+1}|$$

Baker is curious to know the sum of playfulness among all different configurations of arcs (i.e., distinct sets of $p_{1\ldots m}$ and $q_{1\ldots m}$) that satisfy the constraint. As the result can be huge, Baker only requires the answer modulo $10^9 + 7$.

Input

The first line contains two integers n and m ($2 \le n, m \le 250$).

Output

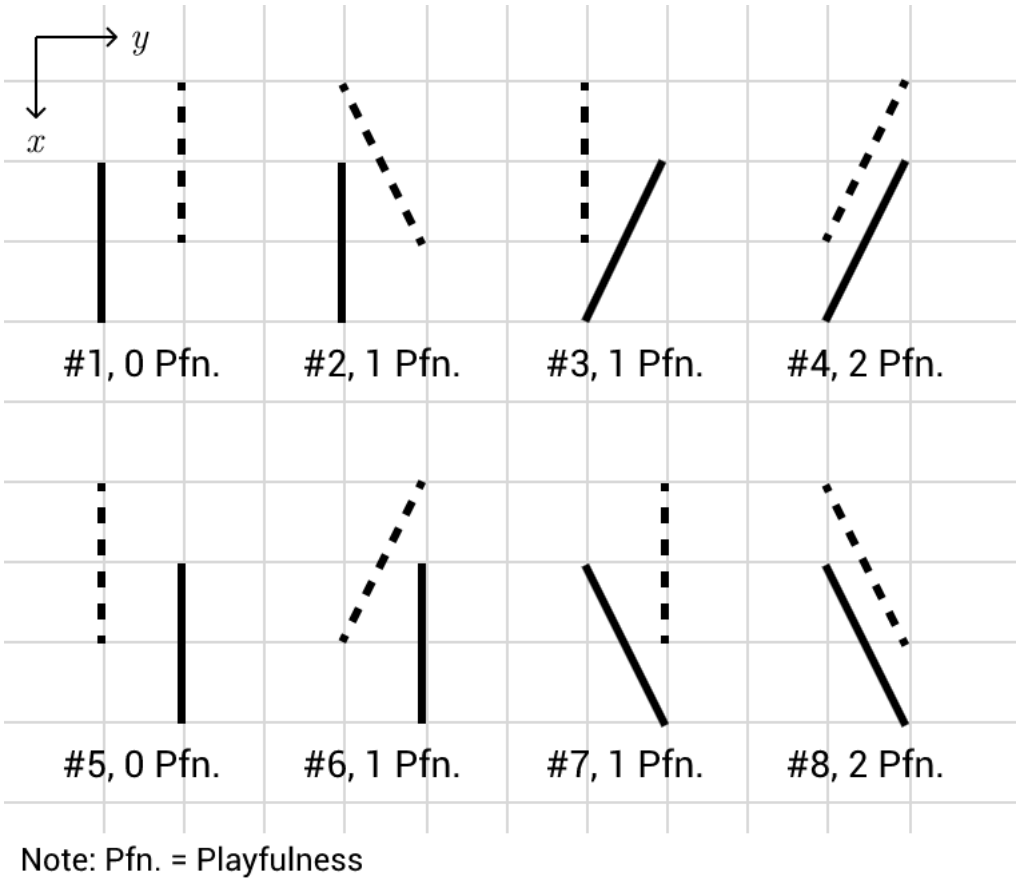
Output a single integer in a line — the answer modulo $10^9 + 7$.

Examples

standard input	standard output
2 2	8
5 3	19716
114 191	24045753

Note

In the first example, there are 8 different configurations of arcs, as shown below.



Note: we rotated the coordinate system to display all the solutions better.

Problem C. Contest Reactions

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

In the SYSUCPC 2077, a total of $k+1$ teams have participated. Unlike previous contests, ACMM Club plans to generate contest reactions based on the scoreboard. Your task here is to list the times when Team 0 solved **new** problems and to document the status of Team 0 when a problem is solved.

There are n submissions for the contest, each of which is described by a combination of the following elements:

- **Timestamp**: this is displayed in the format `hh:mm:ss.SSS`, where `hh`, `mm`, `ss`, `SSS` represent hours, minutes, seconds, and milliseconds, respectively.
- **Submitter ID**: This is an integer between 0 and k.
- **Problem ID**: This can be any uppercase letter, as there are exactly 26 problems in SYSUCPC 2077.
- **Result type**: This can be either `AC` (Accepted) or `RJ` (Rejected).

The penalty for a team is calculated using the following formula:

$$\text{Penalty} = \sum_{\text{solved problems}} \text{FirstSolveTime} + 20 \times \#\text{FailedAttempt}$$

In this formula, FirstSolveTime represents the elapsed minutes of the first accepted submission for the problem since the start of the contest ($= 60 \times \texttt{hh} + \texttt{mm}$), and #FailedAttempt is the number of failed attempts (rejected submissions that occurred **before the first accepted submission**).

Teams are ranked in **descending order** by the number of problems solved. If teams solve the same number of problems, they are sorted in **ascending order** by penalty. To simplify the problem, teams with **the same number of problems solved and penalties** share the same rank number. Additionally, we will assume that the scoreboard is not frozen during the contest.

You can further understand how the ranking system works by observing the subsequent image provided.

Input

The first line contains two integers k and n ($1 \le k \le 10^5$, $1 \le n \le 2 \times 10^5$).

Each of the following n lines describes a submission, which includes the timestamp, submitter ID, problem ID, and result type. The submissions are given in ascending order by timestamp.

It is guaranteed that all timestamps in the input are unique and fall within the range of `00:00:00.000` to `04:59:59.999`.

Output

When Team 0 solved a new problem, output one line in the following format: `hh:mm:ss.SSS X #x -> #y`, denoting that Team 0 solved problem `X` at time `hh:mm:ss.SSS`, resulting in their rank changing from x to y.

The output should be sorted in ascending order by timestamp. **Note that output should be provided even if the rank did not change after solving a new problem.**

Example

standard input	standard output
3 14 00:11:45.140 1 A RJ 00:12:45.140 2 A RJ 00:20:11.111 1 A AC 00:21:22.222 2 A AC 00:41:33.333 0 A AC 00:49:44.444 0 A RJ 00:50:55.555 0 B RJ 01:30:00.666 3 A AC 01:31:11.777 3 B AC 01:32:22.888 3 C RJ 01:40:33.999 1 B AC 02:00:44.000 0 B AC 04:05:55.100 0 C AC 04:59:59.999 0 C AC	00:41:33.333 A #3 -> #2 02:00:44.000 B #3 -> #2 04:05:55.100 C #2 -> #1

Note

The image below displays the scoreboards after Team 0 solved a new problem. The = column in the tables indicates the number of problems solved, and the $\sum$ column indicates the penalty.

Scoreboard after Team 0 solved A

Rank	Team	=	Σ	A		B		C	
1	1	1	40	+1	00:20				
2	0	1	41	+	00:41				
2	2	1	41	+1	00:21				
4	3	0	0						

Scoreboard after Team 0 solved B

Rank	Team	=	Σ	A		B		C	
1	1	2	140	+1	00:20	+	01:40		
2	0	2	181	+	00:41	+1	02:00		
2	3	2	181	+	01:30	+	01:31	-1	
4	2	1	41	+1	00:21				

Scoreboard after Team 0 solved C

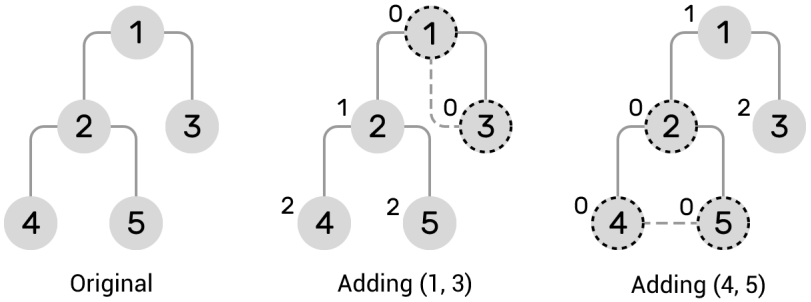
Rank	Team	=	Σ	A		B		C	
1	0	3	426	+	00:41	+1	02:00	+	04:05
2	1	2	140	+1	00:20	+	01:40		
3	3	2	181	+	01:30	+	01:31	-1	
4	2	1	41	+1	00:21				

Problem D. Depths of Cities

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

There are n cities in a country, numbered from 1 to n. Initially, there are $n-1$ bidirectional roads, each connecting two cities, ensuring that all cities can reach each other. In other words, the cities and roads form a **tree** in graph theory.

As the manager of this city, you are planning to construct a subway connecting two cities, u and v (even if the two cities are already connected directly by a road). This subway will form a cycle with some of the existing roads, and the cities on the cycle are called *key cities*. After that, the depth of a city is defined as the minimum distance between the city and any key city. Here, the distance between two cities is determined by the number of roads needed to travel from one to the other. The depth of key cities is defined as zero.



Two plans to construct a subway in the first example. The key cities are marked by dashed stroke, and the depth of each city is given at the top-left of the circle.

There are $\frac{n(n-1)}{2}$ plans for constructing the subway. For each city, what is the total depth across all plans?

Input

The first line contains one integer n ($2 \le n \le 2 \times 10^5$) — the number of cities.

Each of the following $n-1$ lines contains two integers u and v ($1 \le u, v \le n$, $u \ne v$) — a road connecting city u and v. It is guaranteed that cities can reach each other by roads.

Output

Output n lines. The i-th line should contain one integer — the total depth of city i.

Examples

standard input	standard output
5 1 2 1 3 2 4 2 5	3 1 9 7 7
7 1 2 1 3 1 4 2 5 2 6 3 7	4 7 13 19 22 22 28

Problem E. Erasing Numbers

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

Alice and Bob are going to play a game. Initially, there are n numbers lined up on the board, all of which are either 0 or 1. Alice and Bob take turns to move, and Alice moves first.

On Alice's turn, she can perform one of the following actions:

- Choose a number 0 on the board and erase it.
- Choose two numbers 0 on the board and erase them with all the numbers between them.

For example, if the numbers on the board are $0, 1, 1, 0, 1$, Alice can either erase one of the zeros or erase the first four numbers together.

Bob's actions are similar to Alice's, but he should choose 1 instead of 0.

To make the game more interesting, Alice and Bob agree that the first person who is unable to perform any action **wins** (e.g., no number 0 on the board on Alice's turn). If both players play optimally, who will be the winner?

Input

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

Each test case consists of two lines:

- The first line contains one integer n ($1 \le n \le 10^5$) — the initial count of numbers on the board.
- The second line contains a binary string of length n — the n numbers on the board in order.

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

Output

For each test case, output a single line containing `Alice` or `Bob` — the winner if both players play optimally.

Example

standard input	standard output
3 5 10100 5 11011 5 01010	Alice Alice Bob

Problem F. Finding Maxi-strings

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

Mike was doing an everyday challenge on a website. One day, the website gave the following challenge:

You've got a string s consisting of lowercase English letters. Let's define the occurrence number of another string t on s as the number of pairs (l, r) such that $1 \le l \le r \le |s|$ and $s_l s_{l+1} \dots s_r = t$. List the strings with the biggest occurrence number.

Mike figured out the answer within one second. But after he typed all the answers on the website, he noticed that many challengers finished faster than him. Mike thought that the problem was the input order.

The website provides an input area for users to write answers. The input area is initially empty, and the function of it is limited — only the following actions are allowed:

- Adding a lowercase English letter at the end of the input.
- Pressing "Backspace" to remove one letter from the end of the input. This action only works if there are letters in the input area.
- Pressing "Enter" to submit an answer based on the current input. **The content of the input area remains unchanged.**

The challenge finishes when all the answers are submitted. In addition to the original problem, Mike will change the string s for m times, each modification involving a single character. You are required to output the minimum number of actions needed to submit all answers **after every change** to the string s.

Input

The first line contains a single integer n ($1 \le n \le 5 \times 10^5$) — the length of the given string s.

The second line contains a string of n lowercase English letters — the string s.

The third line contains a single integer m ($1 \le m \le 10^5$) — the number of modifications.

Each of the next m lines contains a single integer p ($1 \le p \le n$) and a lowercase English letter ch — a modification specifying that the p-th letter of string s should be changed to ch.

Output

After each modification, output a single integer in a line — the minimum number of actions required to submit all answers.

Examples

standard input	standard output
6 ababab 4 1 c 3 c 6 a 5 c	2 2 10 2
23 fadejademadewademinimum 3 1 j 9 j 13 j	18 15 26

Note

In the first example, the string s becomes `cbcbaa` after the third change. The answers are `a`, `b`, `c`, and `cb`, as each of these appears 2 times in string s. A possible way to submit these answers in 10 actions is as follows:

$$\texttt{[empty]} \xrightarrow{\texttt{Add a}} \texttt{a} \xrightarrow{\texttt{Enter}} \texttt{a}\checkmark \xrightarrow{\texttt{Backspace}} \texttt{[empty]} \xrightarrow{\texttt{Add b}} \texttt{b} \xrightarrow{\texttt{Enter}} \texttt{b}\checkmark$$
$$\xrightarrow{\texttt{Backspace}} \texttt{[empty]} \xrightarrow{\texttt{Add c}} \texttt{c} \xrightarrow{\texttt{Enter}} \texttt{c}\checkmark \xrightarrow{\texttt{Add b}} \texttt{cb} \xrightarrow{\texttt{Enter}} \texttt{cb}\checkmark$$

Problem G. General Checksum Calculation

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

After several years of hard work, a group of students from Sun Yat-sen University has successfully developed their first handcrafted CPU, named **SYS-Unit**. The initial task for SYS-Unit is "General Checksum Calculation".

There are n integers $a_1, a_2, \cdots, a_n$ stored in memory, and SYS-Unit is required to compute k checksums based on these integers. For the i-th checksum, three arguments are provided: d_i, l_i, and r_i. The i-th checksum is formulated as:

$$\oplus_{p=l_i}^{r_i}(a_p - d_i)$$

where $\oplus$ denotes the bitwise XOR operator. To simplify the process, all requests ensure that for every index p within the range $[l_i, r_i]$, the condition $d_i \le a_p$ holds.

You are asked to pre-compute all checksums in order to verify the correctness of SYS-Unit.

Input

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

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i < 2^{17}$).

For the next k lines, the i-th one contains three integers d_i, l_i, and r_i ($0 \le d_i < 2^{17}$, $1 \le l_i \le r_i \le n$). It is guaranteed that for all $l_i \le p \le r_i$, $d_i \le a_p$ holds.

Output

Output k lines. The i-th line should contain one integer — the i-th checksum.

Example

standard input	standard output
7 4 11 45 14 19 19 8 10 1 1 4 5 1 4 1 4 7 14 2 4	57 41 14 26

Note

In the example:

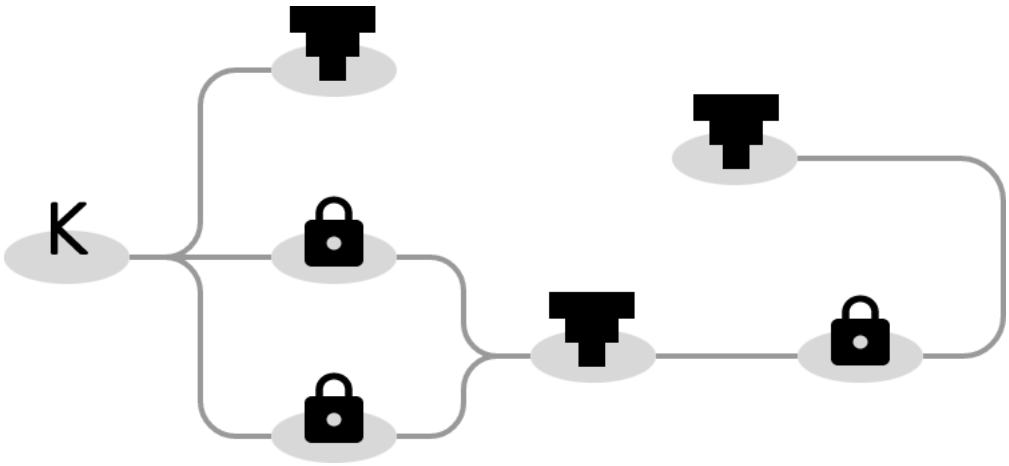
- The first checksum is $10 \oplus 44 \oplus 13 \oplus 18 = 57$;
- The second checksum is $6 \oplus 40 \oplus 9 \oplus 14 = 41$;
- The third checksum is $18 \oplus 18 \oplus 7 \oplus 9 = 14$;
- The fourth checksum is $31 \oplus 0 \oplus 5 = 26$.

Problem H. Hoppers and Doors

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

K was trapped in a Minecraft prison created by a YouTuber named Omz, who had a penchant for using hoppers in his prisons. The hopper in this problem can be viewed as a container that can store **any number of items**.

Fortunately, he soon realized that barrier blocks did not surround the prison, allowing him to use **F5** to see the entire area. The prison could be described as a maze consisting of $n+m+1$ nodes, starting with K at node 1. There were n hoppers, each positioned at a unique node between 2 and $n+1$, and m doors, each located at a unique node between $n+2$ and $n+m+1$. Several bidirectional paths connected two nodes, ensuring that **all nodes can be reached after all doors are opened**. However, K could only enter a node with a door if he had found a way to open it. The final goal for K is to **open all m doors**.



The image above shows the prison specified by the second example.

Based on his intuition, K suspects that there are m keys hidden in the hoppers, with each key corresponding to a unique door. He wonders: if each key is placed in a random hopper, how many of the n^m possible arrangements of keys in hoppers would allow K to collect all the keys and achieve his goal?

Input

The first line contains three integers n, m, and r ($1 \le n, m \le 11$, $n+m \le r \le \frac{(n+m)(n+m+1)}{2}$) – the number of hoppers, doors, and bidirectional paths.

Each of the next r lines contains two integers u and v ($1 \le u < v \le n+m+1$) – a path connecting node u and node v.

It is guaranteed that no two paths connect the same two nodes, and all nodes can be reached from node 1 after all doors are opened.

Output

Output a single integer in a line — the number of possible arrangements of keys in hoppers.

Examples

standard input	standard output
2 2 5 1 2 2 4 2 5 3 4 3 5	3
3 3 7 1 2 1 5 1 6 3 5 3 6 3 7 4 7	10

Problem I. Isla Loves Christmas

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

Christmas is coming! Isla bought n colorful Christmas lights online and strung them up with a wire. There are k different colors of lights, each represented by a distinct integer between 1 and k. The color of the i-th light from the left is denoted as a_i.

Isla plans to decorate the lights with m actions. The i-th action will focus on a continuous section of lights, specifically between the l_i-th and the r_i-th light. The action is defined as follows:

For every pair of positions (p, q) such that $l_i \le p \le q \le r_i$ and $a_p = a_q$, a note with a non-negative integer value v_i will be placed on the x-th light if $p \le x \le q$ and $a_x = a_p$ are satisfied.

After the actions, Isla wants to know the sum of the values on the notes for each light. As the result can be huge, she only cares about the sum modulo $10^9 + 7$.

Input

The first line contains two integers n and k ($1 \le k \le n \le 10^5$) — the number of lights and colors.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le k$) — the color of the lights.

The third line contains one integer m ($1 \le m \le 10^5$) — the number of actions.

For the next m lines, the i-th one contains three integers l_i, r_i, and v_i ($1 \le l_i \le r_i \le n$, $1 \le v \le 10^5$) — the information of the i-th action.

Output

Output one line containing n integers. The i-th integer should equal the sum of the values on the notes for the i-th light modulo $10^9 + 7$.

Example

standard input	standard output
10 3 3 3 1 1 2 1 1 2 2 2 3 1 9 1 3 7 10 5 10 100	2 2 44 66 413 266 244 604 603 400

Problem J. Jazz Music from the Er-th

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

You may have heard about the new planet discovered by humans — The Er-th. The "Er" in its name means two, as creatures on this planet only use binary integers and operations (e.g., `and`, `or`, `xor`).

Scientists have found a way to communicate with these creatures. The discovery indicates that the creatures enjoy jazz music, and they have a formal way of evaluating the beauty of jazz music. Assume that a piece of jazz music can be cut into k slices, each with an integer rhythmicity value R_i that is **not less than a constant value** L. The beauty of this jazz music is then defined as $R_1|R_2|\ldots|R_k$ (where $|$ denotes the bitwise OR operator).

After learning subtraction from human beings, the creatures figure out a new way to compose jazz music — by decreasing some rhythmicity values of a piece of jazz music (while ensuring they remain **not less than** L), the beauty can sometimes increase. Now, they are curious about a problem: What is the maximum beauty of a piece of jazz music that can be achieved by decreasing some (possibly none) rhythmicity values?

Input

The first line contains a single integer T ($1 \le T \le 10^5$) — the number of test cases.

Each test case consists of two lines:

- The first line contains one integer n and L ($1 \le n \le 10^5$, $0 \le L < 2^{60}$) — the number of slices of the music and the constant value.
- The second line contains n integers $a_1, a_2, \ldots, a_n$ ($L \le a_i < 2^{60}$) — the rhythmicity values of the music.

It is guaranteed that the sum of n over all test cases does not exceed 2×10^5. Also, to simplify your research, all numbers in the input will be given in decimal.

Output

For each test case, output a single integer in a line — the maximum beauty the music can achieve. Your output should also be given in decimal.

Example

standard input	standard output
3 3 100 1924 100 2024 4 10 10 34 16 10 4 16 260 128 84 36	2047 59 503

Note

In the second test case, the maximum beauty of 59 can be achieved by decreasing the second rhythmicity value to 33.

In the third test case, the maximum beauty of 503 can be achieved by decreasing each rhythmicity value to 258, 128, 84, and 33, respectively.

Problem K. King of Card Games

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

An innovative card drawing game, developed by K7K7 Games, has become famous among ACMM Club.

In this card game, a deck consists of n cards, each marked with a distinct integer from 1 to n written on them. The player must perform the following action m times: draw a card randomly from a deck, record the number on this card, and then return it to the deck. After m draws, if the maximum and minimum numbers recorded are a and b, respectively, the "level of bad luck" is defined as $(a - b)^2$.

Maple aspires to become the king of card games. She is curious about the total sum of the "level of bad luck" for all n^m possible sequences of card drawings. Since the number can be huge, Maple is only interested in the sum modulo 998 244 353.

Input

The first line contains two integers n and m ($1 \le n, m \le 10^6$) — the number of cards in a deck and the number of card draws.

Output

Output a single integer — the total sum of the "level of bad luck" modulo 998 244 353.

Examples

standard input	standard output
3 1	0
3 2	12
114 514	486446955

Note

In the first example, the "level of bad luck" is always zero since only one number is recorded.

In the second example, there are 9 possible sequences of card drawings.

- Three of the sequences have a "level of bad luck" of 0;
- Four of the sequences have a "level of bad luck" of 1;
- Two of the sequences have a "level of bad luck" of 4.

In summary, the total sum of the "level of bad luck" equals $3 \times 0 + 4 \times 1 + 2 \times 4 = 12$.

Problem L. LCM and GCD

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

As you all know, you are participating in the SYSUCPC (SYSU Constructive Problem Contest) finals. Undoubtedly, the Constructive Kingdom of Sun Yat-sen University is eager for you to tackle some related problems.

Before you challenge the problem from the constructive kingdom, here's another constructive problem for you to warm up.

You are given n integers $a_1, a_2, \ldots, a_n$ and m integers $b_1, b_2, \ldots, b_m$. An n-by-m matrix $mat_{i,j}$ is called *good* if it satisfies all requirements below:

- $1 \leq mat_{i,j} \leq 10^9$ holds;
- The **LCM (Least Common Multiple)** of numbers in the i-th row equals a_i;
- The **GCD (Greatest Common Divisor)** of numbers in the j-th column equals b_j.

Construct a *good* matrix or determine if it is impossible.

Input

The first line contains two integers n, m ($1 \leq n, m \leq 1000$).

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 10^9$).

The third line contains m integers $b_1, b_2, \ldots, b_m$ ($1 \leq b_j \leq 10^9$).

Output

If it's impossible to construct a *good* matrix, output a single line containing `No`.

Otherwise, output `Yes` on the first line, and then output n lines, each containing m numbers, denoting your construction. If there are multiple answers, print any of them.

You can output `Yes` and `No` in any case (for example, strings `yEs`, `yes`, and `YES` will be recognized as a positive response).

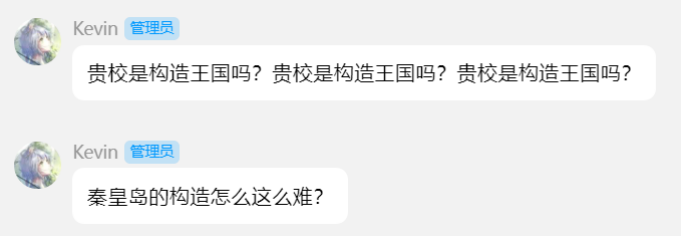
Examples

standard input	standard output
2 3 2 6 1 2 2	Yes 2 2 2 1 6 2
3 3 1 1 4 5 1 4	No

Problem M. Make SYSU Great Again 3

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

As you all know, you are participating in the SYSUCPC (SYSU Constructive Problem Contest) finals. Undoubtedly, the Constructive Kingdom of Sun Yat-sen University is eager for you to tackle some related problems.



Maple has a set of n integers: $1, 2, \ldots, n$ (where $n \geq 4$). She wants to arrange these numbers in a circle. There are n selections to pick three adjacent numbers from the circle, and she wants to find an arrangement such that at least $\lceil n/2 \rceil$ of the selections satisfy the condition where the largest number equals the sum of the other two numbers.

As a friend of Maple, you are asked to either construct such an arrangement or determine that it does not exist.

Input

The first line contains one integer T ($1 \leq T \leq 100$) — the number of test cases.

Each test case contains one integer n ($4 \leq n \leq 2 \times 10^5$).

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

Output

For each test case:

If it's impossible to construct the arrangement, output a single line containing `No`.

Otherwise, output `Yes` on the first line, then output n numbers on the second line, denoting your construction. If there are multiple answers, print any of them.

You can output `Yes` and `No` in any case (for example, strings `yEs`, `yes`, and `YES` will be recognized as a positive response).

Example

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

Note

In the second test case, there are 4 valid selections in the arrangement: $(3, 2, 5)$, $(5, 4, 1)$, $(4, 1, 3)$, and $(1, 3, 2)$, so this arrangement satisfies the requirement of at least $\lceil n/2 \rceil = 3$ valid selections.