2025 KAIST 15th ICPC Mock Competition

Hosted By:

Sponsored By:

Rules

- This contest is 2025 KAIST 15th ICPC Mock Competition.
- This contest is sponsored by Hudson River Trading, Jane Street, Samsung Software Membership, Startlink, and utilForever.
- You can only participate in a team of at most three people.
- Use of the network is prohibited during the competition, except for submitting source codes and accessing language reference sites. Here are the allowed reference sites:
  - `C/C++ : https://en.cppreference.com/w/`
  - `Java : https://docs.oracle.com/en/java/javase/`
  - `Python : https://docs.python.org/`
  - `Kotlin : https://kotlinlang.org/docs/`
- Each team can bring up to 25 pages of printed, A4-sized, single-sided reference materials for use during the competition.
- The contest lasts 5 hours.
- The contest consists of 13 problems.
- Each problem has time limit and memory limit. This means your solution should run in the given time and memory limit for each test. Time limits and memory limits are language-independent.
- Problems may have different time limits.
- The memory limit for every problem is 1024 MiB.
- All programming languages available on Baekjoon Online Judge (BOJ) are allowed.
- Every problem is guaranteed to be solvable using `C++20`. This is not guaranteed for any other languages.
- Solutions to problems submitted for judging are called *runs*. Each run is judged as accepted or rejected, and the team is notified of the results. Rejected runs will be marked with one of the following: Compilation Error, Runtime Error, Time Limit Exceeded, Wrong Answer, Memory Limit Exceeded, Presentation Error, Output Limit Exceeded.
- A problem is solved when it is accepted by the judges.
- Teams are ranked according to the most problems solved. Teams who solve the same number of problems are ranked first by the least total penalty and, if need be, by the earliest time of submission of the last accepted run, except the runs to the problems that is already solved.
- The total penalty is the sum of the penalty for each solved problem. The penalty for a solved problem is the time elapsed from the beginning of the contest to the submission of the first accepted run, rounded down to the minutes, plus 20 penalty minutes for every previously rejected run before the first accepted run for that problem that was not rejected due to Compilation Error. There is no penalty for a problem that is not solved.

Problem list

#	Problem name	Time limit (All languages)
A	Armageddon	1 second
B	Brain Power	1 second
C	Conflict	3 seconds
D	Don't Fight the Music	3 seconds
E	EVANESCENT	3 seconds
F	Freedom Dive	1 second
G	Grievous Lady	1 second
H	Halcyon	10 seconds
I	Impact	2 seconds
J	Judgement	2 seconds
K	Kamui	2 seconds
L	Last Celebration	5 seconds
M	Misdeed -la bonté de Dieu et l'origine du mal-	5 seconds

Problem A. Armageddon

Time limit: 1 second
Memory limit: 1024 megabytes

Cocoa, the *magic sword president* of RUN, has obtained a new magic sword named **Armageddon**. To maximize its power, Cocoa intends to enhance it.

Specifically, the sword's power is determined by its length x, enchantment level y, and brilliance z, according to the formula:

$$\frac{x(x+1)}{2} \cdot \frac{y(y+1)}{2} \cdot a^z$$

where x, y, z are nonnegative integers. Here, a is a fixed constant determined when **Armageddon** was forged. Initially, x, y, z are all initialized to 0.

Cocoa can invest her mana to improve the sword. For each 1 mana spent, she can increase either x, y, or z by 1.

For each $k = 1, 2, \cdots, n$, determine the maximum possible power of the sword if Cocoa uses exactly k mana to enhance the sword.

Input

The first line contains three integers p, q, and n separated by spaces, where $a = p/q$.

Output

Print n values in a single line, separated by spaces.

For the i-th value, output $s \times t^{-1} \pmod{10^9 + 7}$, where s/t is the irreducible fraction representing the maximum possible power after investing exactly i mana.

Constraints

- $1 \leq p, q, n \leq 10^5$

Examples

standard input	standard output
1 1 10	0 1 3 9 18 36 60 100 150 225
2 1 10	0 1 3 9 18 36 72 144 288 576
100000 1 5	0 1 100000 999999937 993000007

Note

There is no *magic sword president* in RUN.

Problem B. Brain Power

Time limit: 1 second
Memory limit: 1024 megabytes

You are given a string s consisting of lowercase English letters. Your task is to split s into a sequence of non-empty **substrings** such that no two adjacent substrings in the sequence are **anagrams** of each other. (Two strings are considered anagrams if they contain the same characters with the same frequencies.) Among all such valid splits, you must maximize the number of substrings.

Input

The first line of the input contains a single integer T, the number of test cases.

The following T lines each describe a test case. Each line contains a single string s consisting of lowercase English letters.

Output

For each test case, print a single integer on a new line: the maximum possible number of substrings in a valid split.

Constraints

- $1 \leq T \leq 10^5$
- $1 \leq |s| \leq 10^5$ for each test case.
- The total length of all strings s over all test cases does not exceed 10^5.

Examples

standard input	standard output
5 kaist rrunnn iiccppcc mooockk conntttest	5 4 6 5 8
8 a bb ccc dddd eeeee ffffff ggggggg hhhhhhhh	1 1 2 3 3 4 5 5
4 brainpowerletthebasskick ooooooooooaaaaeaaiau jooooooooooooaaeoaauua eeeeeeeeeaaaaeaeiea	22 15 17 15

Problem C. Conflict

Time limit: 3 seconds
Memory limit: 1024 megabytes

In a desperate conflict, with a ruthless enemy...

This is an interactive problem.

As an elite spy fighting against a great evil empire, you have been tasked with T separate reconnaissance missions. Each mission takes place in a different city of the empire, and your success is crucial to the resistance.

Every city is represented by N key buildings, numbered from 1 to N, connected by a network of roads. Each road connects two different buildings, and there may be multiple roads between the same pair of buildings. Your objective for each mission is to fully reconstruct the city's road network.

To achieve this, you are equipped with a special device capable of performing a **sequential shutdown** of the city's power grid. You can define a shutdown sequence $p_1, p_2, \ldots, p_N$ — a permutation of all N buildings. At the exact moment when the power to building p_i is cut, the device reports the number of roads that connect p_i to other buildings that **still have power.**

You are allowed to use this device at most $N - 1$ times during each mission.

Interaction Protocol

The first line of the input contains a single integer T, the number of test cases. Then, T test cases follow.

For each test case, the interaction begins with a single integer N, the number of buildings, given to your program via standard input.

Your program can then perform sequential shutdowns to explore the city's road network. To do this, you may issue queries of the form:

- `?` p_1 p_2 $\cdots$ p_N — Initiate a shutdown using your chosen **shutdown sequence** $p_1, \ldots, p_N$.

In response to such a query, a single line containing N space-separated integers $c_1, c_2, \ldots, c_N$ will be given to your program via standard input. Here c_i is the measurement taken at the moment building p_i's power is cut: the number of roads connecting it to other buildings that **still have power**.

You can perform at most $N - 1$ sequential shutdowns per test case.

Once you have completed your mission and mapped out the entire road network, you must report your findings. First, print a line in the following format:

- `!` M

where M is the total number of roads you have discovered.

Then, for the next M lines, print two space-separated integers u and v, representing a discovered road between buildings u and v. If multiple roads exist between the same pair, you must print that pair multiple times. The roads may be printed in any order; any order is acceptable.

The interactor is **non-adaptive**: the road network of each city is fixed in advance and does not depend on your queries.

Constraints

- $1 \leq T \leq 1\,000$
- $2 \leq N \leq 1\,000$
- $0 \leq M \leq 10^4$
- The sum of N over all test cases does not exceed $2\,000$.

- The sum of M over all test cases does not exceed 10^4.

Example

standard input	standard output
2 3 2 1 0 2 1 0 5 5 1 0 0 0 2 3 1 0 0	 ? 1 2 3 ? 3 2 1 ! 3 1 2 2 3 3 1 ? 1 2 3 4 5 ? 5 1 2 3 4 ! 6 1 2 1 3 2 3 1 4 1 5 1 5

Note

After printing each query, you must flush the output buffer to ensure the interactor receives your output. Failing to do so can result in an unexpected verdict. You can flush the output by using the following methods:

- In `C++`, call `fflush(stdout)` or `cout.flush()`.
- In `Java`, call `System.out.flush()`.
- In `Python`, call `sys.stdout.flush()`.
- In `Kotlin`, call `System.out.flush()`.

For other languages, you should refer to the official documentation for your language.

Problem D. Don't Fight The Music

Time limit:	3 seconds
Memory limit:	1024 megabytes

There are N cards in a row. Each card has a red front side and a blue back side. An integer R_i is written on the red side of the i-th card, and an integer B_i is written on the blue side. Initially, every card is facing either red side up or blue side up.

An operation on a range $[l, r]$ is defined as follows:

- For each i from l to r, let c_i be the number of indices j with $l \leq j < i$ such that card j shows the same color as card i at the beginning of the current operation.
- After the operation, card i shows its red side if c_i is even, and its blue side if c_i is odd.
- All cards in $[l, r]$ are updated simultaneously.

You need to process the following Q queries:

- `1 i` — Flip the i-th card.
- `2 i k` — Change the value on the red side of the i-th card to k.
- `3 i k` — Change the value on the blue side of the i-th card to k.
- `4 l r T` — Calculate the sum of the numbers on the face-up sides in the range $[l, r]$, when the operation has been applied to this range T times. The state of the cards does not change as a result of this query.

Input

The first line contains a single integer N, the number of cards.

The second line contains a string s of length N, consisting of characters `R` and `B`. The i-th character of s denotes the initial side of the i-th card (`R` for red side up, `B` for blue side up).

The third line contains N integers $R_1, R_2, \cdots, R_N$ — the values on the red sides of the cards.

The fourth line contains N integers $B_1, B_2, \cdots, B_N$ — the values on the blue sides of the cards.

The fifth line contains a single integer Q, the number of queries.

Each of the next Q lines describes a query of one of the four types as described above.

Output

For each query of type 4, output a single integer, the calculated sum for the specified range.

Constraints

- $1 \leq N, Q \leq 2 \cdot 10^5$
- $1 \leq R_i, B_i \leq 10^9$
- For a query of type 1: $1 \leq i \leq N$
- For a query of type 2 or 3: $1 \leq i \leq N$, $1 \leq k \leq 10^9$
- For a query of type 4: $1 \leq l \leq r \leq N$, $1 \leq T \leq 10^9$
- It is guaranteed that there is at least one query of type 4.

Examples

standard input	standard output
5 RRRRR 1 2 3 2 1 5 4 3 4 5 4 4 1 5 1 4 1 5 2 1 2 4 1 4 1	13 11 8
8 RBRBRBRB 451 79 882 122 1289 242 2459 262 697 1082 1888 3070 225 410 751 1089 11 1 5 4 1 6 10121 1 3 2 6 803 3 3 741 4 2 7 11104 1 5 3 8 690 2 5 137 3 6 148 4 3 8 20915	7187 5810 6333

Problem E. EVANESCENT

Time limit: 3 seconds
Memory limit: 1024 megabytes

Reiuji Utsuho and Hakurei Reimu decided to compete to see whose abilities were superior.

Their duel takes place on a grid of size $(N+2) \times (N+2)$. Columns are numbered from 0 to $N+1$ from left to right, and rows are numbered from 0 to $N+1$ from top to bottom. The cell in row i and column j is denoted as (i, j).

Utsuho can use her ability to cause explosions in certain cells of the grid. There is no limit on the number of cells she can choose, but explosions are only allowed in the inner $N \times N$ cells. More precisely, if an explosion occurs at cell (i, j), then its row and column indices must satisfy $1 \leq i, j \leq N$.

Uniquely, the explosions Utsuho uses in this battle are special — Their destructive power grows stronger the farther away a cell is from the explosion's origin. Formally, the damage a cell (x, y) receives from a single explosion at (i, j) is defined as $\max(|x-i|, |y-j|)$. If there are multiple explosions, the total damage to (x, y) is the sum of contributions from all explosions.

Before Utsuho even unleashed her power, Reimu used her foresight. She managed to foresee the total damage that each cell on the grid would receive after all explosions. With this knowledge, she hoped to pinpoint the exact locations of Utsuho's explosions and stop them in advance. However, her vision revealed only the damage values and not the precise origin of the blasts.

Given the total damage values of all cells, find any valid set of explosion locations that could explain Reimu's foresight.

Input

The first line contains a single integer N.

The next $N+2$ lines each contain $N+2$ non-negative integers separated by spaces. The j-th integer on the next i-th line represents the total damage value of cell $(i-1, j-1)$, where rows and columns are numbered from 0 to $N+1$ as described above.

Output

Over a total of N lines, output N integers per line, separated by spaces. Each integer must be either 0 or 1. If the integer in row i and column j is 1, an explosion is scheduled to occur at cell $(i-1, j-1)$. If it is 0, no explosion is scheduled at cell $(i-1, j-1)$.

If multiple possible explosion scenarios exist, outputting just one is sufficient. At least one possible explosion scenario is guaranteed to exist.

Constraints

- $1 \leq N \leq 1\,000$

Example

standard input	standard output
3 2 2 2 2 2 2 1 1 1 2 2 1 0 1 2 2 1 1 1 2 2 2 2 2 2	0 0 0 0 1 0 0 0 0

Problem F. Freedom Dive

Time limit: 1 second
Memory limit: 1024 megabytes

On a scenic coastline, there are N skyscrapers arranged in a line. For each building i (from 1 to N), we know its distance from the sea, L_i, and its height, H_i. We can model the top of each building as a point in a 2D plane at coordinates (L_i, H_i). The buildings are sorted by their distance from the sea, so it's guaranteed that $L_i < L_{i+1}$ for all $1 \le i < N$.

You are a professional skydiver and have planned a spectacular dive for Q different days. On the i-th day ($1 \le i \le Q$), you are given a planned dive location, which is a horizontal coordinate d_i. It is guaranteed that no building is located exactly at d_i.

To prepare for the i-th day's dive, you must perform the following setup:

- First, choose two buildings: one building l located to the left of your dive location (where $L_l < d_i$) and one building r located to the right (where $L_r > d_i$). It is guaranteed that such a pair of buildings always exists.
- Next, connect the tops of these two buildings, i.e., points (L_l, H_l) and (L_r, H_r), with a straight rope.
- Finally, you will make your jump from the point on this rope that is precisely at the horizontal coordinate d_i.

Being a cautious professional, you want to minimize the risk associated with high altitudes. Therefore, for each dive, you must choose the pair of buildings (l, r) that results in the **lowest possible altitude** for the rope at your jump-off coordinate d_i.

Note that the rope is an idealized line segment. It is allowed to pass through or intersect with other buildings; its path is determined only by the two chosen endpoints.

For each of the Q planned dives, find this minimum possible altitude.

Input

The first line contains a single integer N — the number of buildings.

The next N lines describe the buildings. The i-th of these lines contains two integers, L_i and H_i — the distance from the sea and the height of the i-th building. It is guaranteed that $L_1 < L_2 < \cdots < L_N$.

The next line contains a single integer Q — the number of planned diving days.

The next Q lines describe the planned dives. The i-th of these lines contains a single integer d_i — the horizontal coordinate for that day's dive. It is guaranteed that d will not be equal to any L_i.

Output

For each of the Q dives, output a single line containing two space-separated integers, s and t. These two integers must represent the minimum possible starting altitude as an **irreducible fraction** s/t. If the denominator is 1, you should still print it.

Constraints

- $2 \le N \le 2 \cdot 10^5$
- $1 \le Q \le 2 \cdot 10^5$
- $1 \le L_i, H_i \le 10^9$
- $L_1 < d_i < L_N$ $(1 \le i \le Q)$

Examples

standard input	standard output
4 1 4 4 3 7 5 11 2 7 2 3 5 6 8 9 10	11 3 10 3 20 7 19 7 17 7 16 7 15 7
4 1 1 3 3 5 5 7 7 3 2 4 6	2 1 4 1 6 1

Problem G. Grievous Lady

Time limit: 1 second
Memory limit: 1024 megabytes

You are given a grid of size $N \times M$. Your task is to color each cell of the grid with one of four colors: 1, 2, 3, or 4.

There is only one rule: any two adjacent cells must have different colors. Two cells are considered adjacent if they share a common edge.

Some cells in the grid may already be colored. These pre-colored cells are located only on the border of the grid. You must color all the remaining empty cells to create a complete grid that satisfies the rule.

Input

The first line of the input contains a single integer T, the number of test cases.

The first line of each test case contains two integers N and M.

The next N lines describe the initial state of the grid. Each line contains M space-separated integers. A value of 0 represents an empty cell, while values from 1 to 4 represent a cell colored with that specific color.

Output

For each test case, output N lines representing the completed grid.

Each line should contain M space-separated integers, where each integer is a color from 1 to 4.

If multiple solutions exist, you may print any one of them.

Constraints

- $1 \leq T \leq 8\,000$
- $5 \leq N, M \leq 2 \cdot 10^5$
- The sum of $N \times M$ over all test cases does not exceed $2 \cdot 10^5$.
- In the initial grid, any non-zero cells are located only on the border (the first or last row, or the first or last column)
- It is guaranteed that a solution always exists for the given input.

Example

standard input	standard output
2 5 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 7 7 1 0 0 2 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4 0 0 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 0 0 2 0 0 1	1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 1 2 1 2 1 2 3 2 1 2 1 2 3 1 1 2 1 2 1 4 2 4 1 2 1 2 1 4 1 2 1 2 1 2 1 2 3 2 1 2 1 2 3 1 3 2 1 2 1

Problem H. Halcyon

Time limit:	10 seconds
Memory limit:	1024 megabytes

Minseok and Martin have two weighted trees T_1, T_2. They share the same vertex set of size n, where we index each vertex with integers from $1, 2, \ldots, n$.

For a given k, Minseok selects k edges from T_1, and Martin selects $n-1-k$ edges from T_2. The union of their selected edges should form a tree. If this is possible, they should minimize the total weight of selected edges.

Input

In the first line, a single integer N denoting the number of vertices in both trees is given.

In the next $N-1$ lines, description of the first tree is given. Each of the $N-1$ lines contains three integers S_i, E_i, W_i, which indicates there is an edge connecting two vertices S_i, E_i with weight W_i.

In the next $N-1$ lines, description of the second tree is given in the same format.

Output

For all $0 \le k \le n-1$, print the minimum total weight, or print -1 if it is impossible.

Constraints

- $2 \le N \le 250\,000$
- $1 \le S_i, E_i \le N, 1 \le W_i \le 10^9$ $(1 \le i \le N-1)$

Examples

standard input	standard output
5 1 2 10 2 4 20 3 4 30 4 5 50 1 2 15 1 3 25 1 4 35 1 5 25	100 85 80 85 110
9 5 7 6577 4 5 8869 5 9 9088 2 1 124 6 2 410 2 8 8154 4 8 4810 3 4 4268 3 9 763 6 2 8959 7 4 7984 3 8 504 8 6 9085 5 2 4861 1 9 8539 1 7 7834	48529 39568 31019 26748 25491 25661 29669 33975 42300

Problem I. Impact

Time limit:	2 seconds
Memory limit:	1024 megabytes

In the year 3025, RUN is hosting the 1015th KAIST ICPC Mock Competition! To give the N participants a fresh impact, the organizers have prepared a total of N different flavors of pudding for them. Since each participant has a unique taste preference distinct from others, the flavor each person desires is different and corresponds to one of the N flavors prepared by the organizers. For each of the N flavors, there are exactly 2 puddings available, resulting in a total of $2N$ puddings.

The organizers of this year wanted to display the puddings as beautifully as possible. Therefore, they prepared a total of 2 special containers to hold the puddings. Each container is designed to stack puddings in layers, with each container able to hold up to $N + 1$ puddings. The organizers want the flavors of the puddings in the first container to be $1, 2, \cdots, N$ from the bottom up, and the flavors of the puddings in the second container to also be $1, 2, \cdots, N$ from the bottom up.

The organizers asked the AI to fulfill this request, but the AI ignored the flavor conditions and randomly placed $2N$ puddings into each container! Therefore, the organizers wish to arrange the puddings as desired using the following operation.

1. Select a container A containing at least one pudding and a container B capable of holding at least one additional pudding. Note that even if $A = B$, B must have space for at least one more pudding.
2. Move the pudding at the bottommost of container A to the topmost of container B. After this, the pudding that was originally the ith from the bottom in container A becomes the $(i - 1)$-th from the bottom.

The staff has limited time, so they want to perform the given operation no more than $200,000$ times to place all puddings into containers by flavor. Help the staff by writing a program that outputs a method to perform the operations.

Input

The first line contains a positive integer N.

The next two lines contain the flavors of pudding in each container, separated by spaces. The first number n_i on the ith line represents the number of puddings in the ith container. Following the first number on the ith line, n_i integers are given. The jth number among these represents the flavor of the jth pudding from the bottom of the ith container.

Output

The first line outputs the number of operations M to be performed.

The next M lines each output two integers A and B. This signifies performing an operation where the pudding on bottom of the A-th container is moved to the top of the B-th container. It must hold that $1 \leq A, B \leq 2$. Container A must have contained at least one pudding, and container B must not have been completely full of puddings.

After all operations are complete, each container must have puddings arranged so that their flavors are $1, 2, \cdots, N$ from the bottom.

Constraints

- $1 \leq N \leq 100$

Examples

standard input	standard output
1 2 1 1 0	9 1 2 1 2 2 1 2 1 1 2 1 2 2 1 1 2 2 1
2 2 2 1 2 2 1	6 1 1 2 2 1 2 2 1 1 2 2 1

Note

The number of operations need not be minimized.

Problem J. Judgement

Time limit: 2 seconds
Memory limit: 1024 megabytes

You were preparing for the KAIST RUN ICPC 2025 Contest and completely forgot to submit your calculus homework. Now the professor has declared Judgement Day.

He has unleashed the legendary Hubo robot, programmed with only one mission: find you anywhere on campus and gently encourage you to finish the assignment.

The KAIST campus can be modeled as n buildings connected by $n-1$ roads, forming a tree. Each road (u, v) has a **road congestion** $w(u, v)$.

When Hubo is at place x, it picks its next move randomly. For each neighbor y of x, the probability that Hubo moves from x to y is

$$\Pr[x \to y] \;=\; \frac{\frac{1}{w(x,y)}}{\sum\limits_{z \sim x} \frac{1}{w(x,z)}},$$

where $w(x, y)$ is the congestion of road (x, y) and $z \sim x$ denotes all neighbors of x.

Unfortunately for you, the professor can alter the campus roads over time, changing their loads.

You must process q events:

- `1 id new_w` — the congestion of the road with index `id` becomes `new_w`;
- `2 u v` — you are hiding at place v, and Hubo is deployed at place u. Output the **expected number of steps** until Hubo reaches you. If $u = v$, the answer is 0.

Since the expectation can be rational, output it modulo $10^9 + 7$. Formally, if the expectation equals s/t in lowest terms, print $s \times t^{-1} \pmod{10^9 + 7}$, where t^{-1} is the modular inverse of t modulo $10^9 + 7$.

Input

The first line contains two integers n and q. Each of the next $n-1$ lines contains three integers a, b, w describing a road between places a and b with congestion w ($1 \le a, b \le n$). Roads are indexed $1, \cdots, n-1$ in the order they appear. Each of the following q lines is one event in the formats above.

Output

For each query of type 2, print a single integer — the expected number of steps modulo $10^9 + 7$.

Constraints

- $3 \le n, q \le 2 \cdot 10^5$
- For every road congestion w, $1 \le w \le 10^9$

Example

standard input	standard output
4 7 1 2 1 2 3 1 3 4 1 2 1 4 2 4 1 1 2 2 2 1 4 1 1 5 2 4 1 2 3 3	9 9 10 22 0

Problem K. Kamui

Time limit: 2 seconds
Memory limit: 1024 megabytes

There is a graph with a total of $2N$ vertices. There are no edges between the 1st and N-th vertices, and there are no edges between the $(N+1)$-th and $2N$-th vertices. That is, the given graph is a bipartite graph.

A sequence of positive integers $a_1, a_2, \cdots, a_N$ is given. For any (i, j) pair with $1 \leq i, j \leq N$, the necessary and sufficient condition for vertices i and $N+j$ to be connected is that $j \leq a_i$.

A total of Q queries are given. Each query is represented by two integers v and x, indicating that the value of a_v will be changed to $a_v + x$. It is guaranteed that $x = 1$ or $x = -1$. For each query, you must count the number of cycles of length 4 in the given graph. Since the count may be large, output the remainder when divided by 998 244 353. Two cycles are considered different if the sets of edges composing them are different.

Input

The first line contains two positive integers N and Q, separated by a space.

The second line contains a total of N integers $a_1, a_2, \cdots, a_N$, separated by spaces.

The next Q lines each contain two integers v and x separated by a space. The input on the ith line indicates that a_v will be changed to $a_v + x$.

Output

After each query is executed, output the remainder when the number of cycles of length 4 is divided by 998 244 353 on each line.

Constraints

- $2 \leq N \leq 5 \cdot 10^5$, $1 \leq Q \leq 5 \cdot 10^5$
- For each queries, $1 \leq v \leq N$ and $x \in \{-1, 1\}$.
- After each queries, it is guaranteed that $0 \leq a_i \leq N$ for all $1 \leq i \leq N$.

Example

standard input	standard output
10 10 8 6 1 3 0 0 6 9 6 1 6 1 10 -1 5 1 7 -1 7 -1 9 -1 8 -1 7 -1 7 -1 5 -1	178 178 178 158 142 127 127 115 105 105

Note

The set of four edges $\{xy, yz, zw, wx\}$ in a graph is considered to be a cycle of length 4.

Problem L. Last Celebration

Time limit: 5 seconds
Memory limit: 1024 megabytes

To commemorate the end of a memorable era, the city is holding a **Last Celebration**. As a grand finale, a group of N artists has been commissioned to create a final, collaborative mural on a massive city wall. The wall has a length of D and is divided into D sections, numbered from 1 to D. Before they begin, the entire wall is primed with a base color, represented as color 0.

Each artist i is assigned a specific task: to paint the sections from l_i to r_i with their designated color c_i. As the artists work, some sections might be painted over multiple times. The final color of any section is determined by the last artist to paint it.

The quality of the final artwork is determined by its **diversity**. A **block** is defined as a maximal contiguous segment of the wall painted in a single color. The **diversity** of the wall is the total number of blocks.

The N artists will complete their tasks in a random order. Each of the $N!$ possible permutations is equally likely. Your goal is to calculate the **expected value** of the wall's diversity after all artists are finished.

Input

The first line contains two integers, D and N — the length of the wall and the number of artists.

The following N lines each contain three integers, l_i, r_i, and c_i — the range and color for the i-th artist.

Output

Output the expected value of the wall's diversity. Since the expectation can be rational, output it modulo 998 244 353. Formally, if the expectation equals s/t in lowest terms, print $s \times t^{-1} \pmod{998\,244\,353}$, where t^{-1} is the modular inverse of t modulo 998 244 353.

Constraints

- $1 \leq D \leq 10^9$
- $1 \leq N \leq 2 \cdot 10^5$
- $1 \leq l_i \leq r_i \leq D$
- $1 \leq c_i \leq N$

Examples

standard input	standard output
5 2 1 4 1 2 3 2	3
6 3 2 3 1 4 5 1 1 6 2	332748120
1000000000 9 1 2 1 11 22 1 111 222 1 1111 2222 1 11111 22222 1 111111 222222 1 1111111 2222222 1 11111111 22222222 1 111111111 222222222 1	18
4 10 1 1 1 1 2 2 1 3 3 1 4 4 2 2 5 2 3 6 2 4 7 3 3 8 3 4 9 4 4 10	356515843

Problem M. Misdeed -la bonté de Dieu et l'origine du mal-

Time limit:	5 seconds
Memory limit:	1024 megabytes

This is a two-step problem.

Tairitsu and Chunipenguin join forces to overcome the Trial of Karma. The Trial of Karma is structured as follows.

1. Tairitsu receives a binary sequence S of length 196. Subsequently, Tairitsu constructs a 13×13 matrix A where each element is between 0 and 15.
2. The Trial of Karma involves selecting positive integers $1 \leq r_1 < r_2 < \ldots < r_7 \leq 13$ and $1 \leq c_1 < c_2 < \ldots < c_7 \leq 13$. Then create a 7×7 matrix B such that the ith row and jth column of B is A_{r_i,c_j}. Chunipenguin is given this matrix B and the values $r_1, r_2, \ldots, r_7, c_1, c_2, \ldots, c_7$.
3. Chunipenguin must determine S by examining B.

Let's help Tairitsu and Chunipenguin overcome the trials of karma!

Input

The first line contains an integer t that is either 0 or 1. If $t = 0$, it means you must execute Tairitsu's strategy; if $t = 1$, it means you must execute Chunipenguin's strategy.

If $t = 0$, the next line contains 196 integers, either 0 or 1, given without spaces. This represents the binary sequence S received by Tairitsu.

If $t = 1$, a total of 7 positive integers are given over two lines. The ith integer on the first line is r_i, and the ith integer on the second line is c_i.

Subsequently, over the next 7 lines, 7 integers are given per line, separated by spaces. The jth element on the ith line represents B_{ij}.

Output

If $t = 0$, the program must output a total of 13 lines, each containing 13 numbers separated by spaces, where the jth element on the ith line represents A_{ij}.

If $t = 1$, the program must output a total of 196 integers, either 0 or 1, on a single line without spaces. This must be the binary sequence S that Tairitsu initially received.

Examples

standard input	standard output
0 000...000	0 0
1 1 2 3 4 5 6 7 1 2 3 4 5 6 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	000...000