2024 KAIST 14th ICPC Mock Competition

Hosted By:

Sponsored By:

Rules

- This contest is 2024 KAIST 14th ICPC Mock Competition.
- This contest is sponsored by Samsung Software Membership, Moloco, Jane Street, Sorcerics, Startlink, Vieworks, 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.
- The available languages are `C11`, `C++20`, `Java 15`, `PyPy3`, and `Kotlin (JVM)`.
- 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	Automata Embedding	1 second
B	Construct a Coin Set	1 second
C	Counting Regions	2 seconds
D	Graceful Triangles	2 seconds
E	Hexagonal Tiling	4 seconds
F	Jenga Game	1 second
G	Make RUN Great Again	1 second
H	Mosaic	2 seconds
I	Mukjjippa	2 seconds
J	Running in the Plane	1 second
K	Same Segment	1 second
L	Simple Tree Decomposition Problem	3 seconds
M	White-Black-Tree	2 seconds

Problem A. Automata Embedding

Time limit: 1 second
Memory limit: 1024 mebibytes

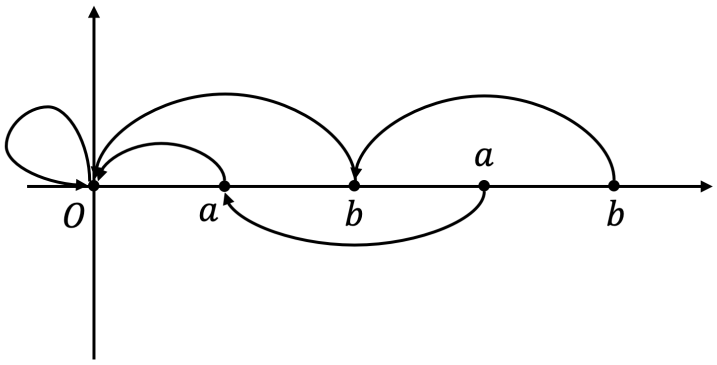
For a string S of length n, let $S[a..b]$ denote the substring consisting of the characters from position a to position b (where $1 \le a \le b \le n$). Also, the failure function $f : [0, n] \to [0, n-1]$ of S is defined as follows.

$$f(i) = \max(\{0\} \cup \{j \mid S[1..j] = S[i-j+1..i],\ 1 \le j < i\})$$

The **KMP automaton** made using the failure function of string S denotes the following kind of automaton. The automaton has $n+1$ **states** $[0..n]$, and for each state $0 \le i \le n$, there exists exactly one **transition** from i to $f(i)$.

If a KMP automaton can be embedded on a plane, it means that if we map state i to a point at $(i, 0)$ on the plane, and draw all transitions $i \to f(i)$ as arrows which do **not** cross the x-axis on the plane, it is possible to draw all $n+1$ arrows such that no arrows intersect except when they meet at endpoints.

Using an alphabet consisting of C letters, find the number of strings of length n whose KMP automaton can be embedded on a plane modulo 998 244 353.



KMP automaton for the string $S =$ `abab`

Input

The first line of input contains two space-separated integers n and C, denoting the length of the string and the number of letters in the alphabet respectively.

Output

The first line of output should contain the number of strings of length n whose KMP automaton can be embedded on a plane modulo 998 244 353.

Constraints

- $1 \le n \le 10^{18}$
- $1 \le C \le 10^{9}$

Examples

standard input	standard output
3 3	27
1000000000000000000 1000000000	609226805

Problem B. Construct a Coin Set

Time limit: 1 second
Memory limit: 1024 mebibytes

The change-making problem is the problem of finding the minimum number of coins that add up to the change to be returned after purchasing items at a shop. A coin set refers to a collection of coin values available for making the change. The greedy algorithm for the change-making problem repeatedly selects the largest coin value among the coin set that does not exceed the remaining amount of change. This process continues until the total amount of change is made.

An optimal solution refers to a solution that uses the fewest number of coins. However, the greedy algorithm does not always guarantee an optimal solution. Under certain conditions, the greedy algorithm may give a non-optimal solution.

Given a positive integer N, your task is to find a coin set for which the greedy algorithm returns an optimal solution for all amounts from 1 won to $(N-1)$ won, but not for N won. Notice that the coin whose value is 1 is always in the coin set.

Input

The first line of input contains the number of test cases T. Each of the next T lines of input contains a single integer N.

Output

For each test case,

If it is impossible to construct a coin set satisfying the conditions given in the problem, print -1.

If it is possible to construct a coin set, print the size of the coin set K in the first line. In the next line, print the coin values $a_1, a_2, \ldots, a_K$, separated by spaces in increasing order. The size of the coin set does not have to be a minimum.

Constraints

- $1 \leq T \leq 1\,000$
- $1 \leq N \leq 10^9$
- $1 \leq K \leq 30$
- $1 = a_1 < a_2 < \cdots < a_K \leq 10^9$

Example

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

Problem C. Counting Regions

Time limit:	2 seconds
Memory limit:	1024 mebibytes

You are given an $N \times N$ grid. A cell in the i-th row and j-th column is denoted as (i, j). Initially, every cell on the grid is colored white.

We will color each cell using $2N-2$ operations. The i-th operation is denoted as (d_i, x_i, c_i). An operation of form (d, x, c) indicates the following:

- For $d = 0$, we color cells in the x-th column. For $d = 1$, we color cells in a x-th row.
- For $c = 0$, we color the column/row white. For $c = 1$, we color the column/row black.

It is guaranteed that if you carry out the operation in the order, each of the $2, 3, 4, \ldots, N$-th row will be colored **exactly once in that order**, and each of the $2, 3, 4, \ldots, N$-th column will be colored **exactly once in that order**. Note that no operation colors the first row and the first column. Formally, the following holds:

- For all integers i, j such that $0 \le i \le 1$ and $2 \le j \le N$, there is a unique integer $1 \le k \le 2N-2$ such that $(d_k, x_k) = (i, j)$.
- For all integers i, j such that $1 \le i < j \le 2N-2$ and $d_i = d_j$, $x_i < x_j$ holds.

A region is defined as maximal sections of neighboring cells of the same color, where two cells are considered neighbors if they share an edge. You need to find the number of regions after performing each $2N-2$ operation in the given order.

Of course, this problem is too easy, so we prepared Q queries for you! Each query is denoted as 3 integers (z, l, r). After the query, you should set $c_i = 1 - c_i$ for all i-th operation where $l \le x_i \le r, d_i = z$ holds. Then, with the changed operation sequence, you need to find the number of regions after performing each $2N-2$ operation. Note that the queries are cumulative.

Input

The first line contains two space-separated integers N and Q.

The i-th line of the next $2N-2$ lines contains three space-separated integers d_i, x_i, c_i.

The i-th line of next Q lines contains three space-separated integer z, l, r, describing the i-th query.

Output

After each query, output a line with a single integer, which is the number of regions.

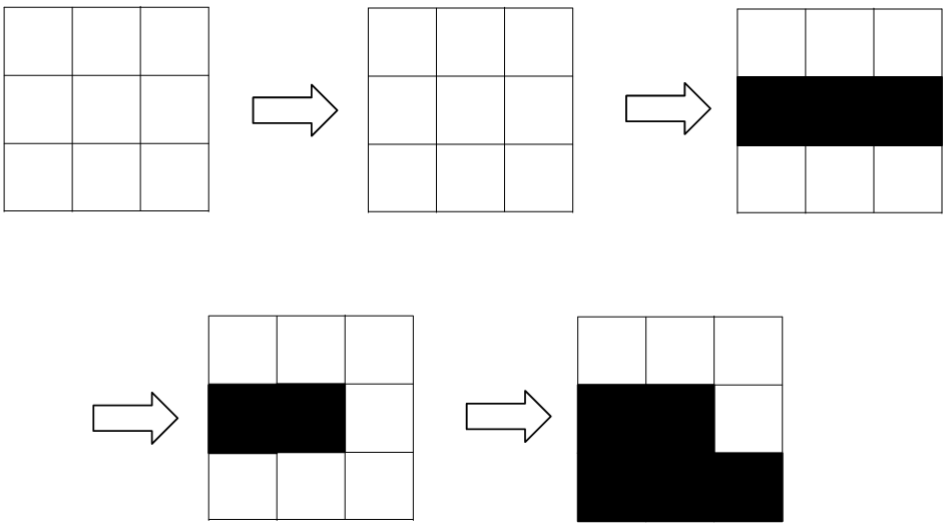
Constraints

- $2 \le N, Q \le 2 \times 10^5$
- For each operation, $0 \le d_i, c_i \le 1$ and $2 \le x_i \le N$
- For all integers i, j such that $0 \le i \le 1$ and $2 \le j \le N$, there is a unique integer $1 \le k \le 2N-2$ such that $(d_k, x_k) = (i, j)$.
- For all integers i, j such that $1 \le i < j \le 2N-2$ and $d_i = d_j$, $x_i < x_j$ holds.
- $0 \le z \le 1$; $2 \le l \le r \le N$ for each query.

Examples

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

Note

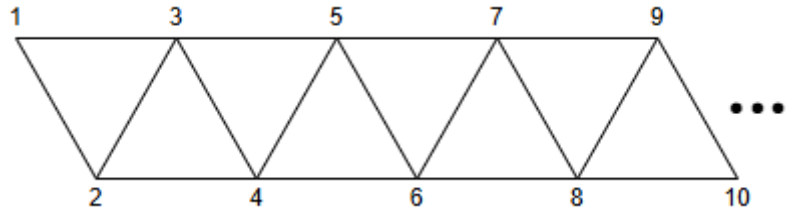


State of the grid after each operation for example 2

Problem D. Graceful Triangles

Time limit: 2 seconds
Memory limit: 1024 mebibytes

Consider the following graph in the shape of n equilateral triangles stitched together horizontally:



This graph has $n+2$ vertices and $2n+1$ edges. The vertices are labeled in the order of increasing horizontal position, as in the image above.

In other words, the graph has $n+2$ vertices labeled from 1 through $n+2$, and $2n+1$ edges connecting all pairs of vertices whose labels differ by at most 2.

A positive integer value is assigned to each vertex. Vertex i has the value of v_i. The value of an edge that connects vertices i and j is $|v_i - v_j|$. Find a way to assign values to all vertices so that for every positive integer k up to $2n+1$ inclusive, exactly one edge has the value of k. The value of any vertex cannot exceed 10^{18}.

Input

The first line contains n, a positive integer.

Output

If a solution exists for the given n, print the values assigned to the vertices $1, 2, \ldots, n+2$ in one line, separated by spaces. The values must be positive integers not exceeding 10^{18}. Otherwise, print -1.

Constraints

- $1 \leq n \leq 200\,000$

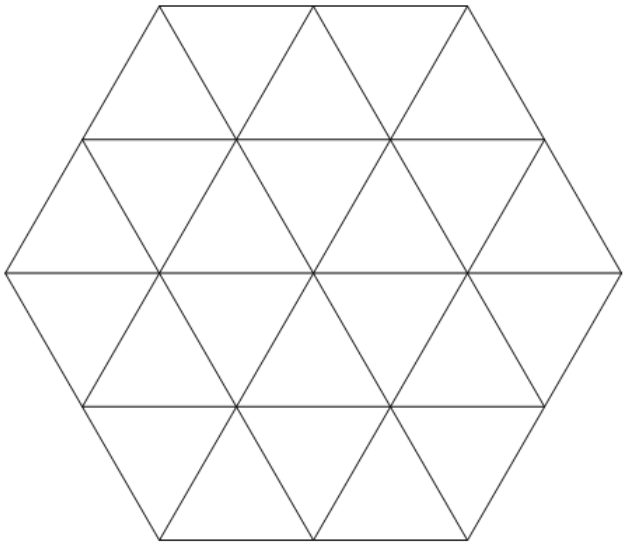
Example

standard input	standard output
1	3 1 4

Problem E. Hexagonal Tiling

Time limit:	4 seconds
Memory limit:	1024 mebibytes

You are given a regular hexagon having sides of length N. A regular hexagon can be split into unit equilateral triangles of side length 1 as shown in the figure below. We are going to completely fill the hexagon with unit rhombuses of side length 1 formed by joining two equilateral triangles which share an edge.



Hexagon formed from triangles

For each position a unit rhombus can be placed, the cost of placing a rhombus is given. Find the minimum cost required to fill the hexagon.

Input

The first line of input contains N.

The following $2N$ lines contain the cost for a rhombus placed in each respective row.

Let's say the cost of a rhombus formed by joining the j-th and $j+1$-th triangles of the i-th row is $p_{i,j}$.

The i-th of the $2N$ lines of input contains $p_{i,1}, p_{i,2}, \ldots$.

The next $2N-1$ lines of input contain the cost for a rhombus placed across two rows.

Let's say the cost of a rhombus formed by joining the j-th inverted triangle of the i-th row and the triangle above it is $q_{i,j}$.

The i-th of the $2N-1$ lines contains $q_{i+1,1}, q_{i+1,2}, \ldots$.

Output

Print the minimum cost required to fill the hexagon using unit rhombuses. It can be proved that it is always possible to fill a hexagon using unit rhombuses.

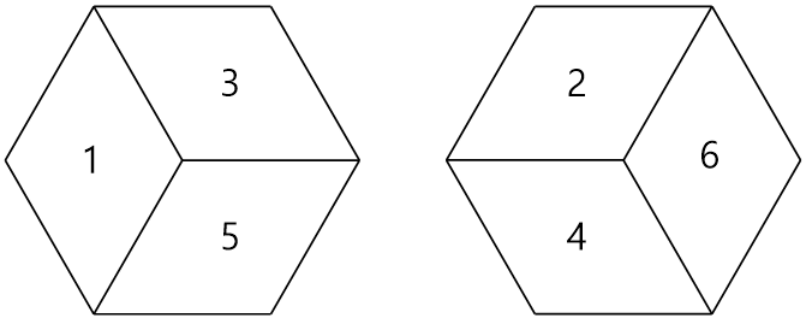
Constraints

- $1 \leq N \leq 100$
- $0 \leq p_{i,j}, q_{i,j} \leq 10^9$

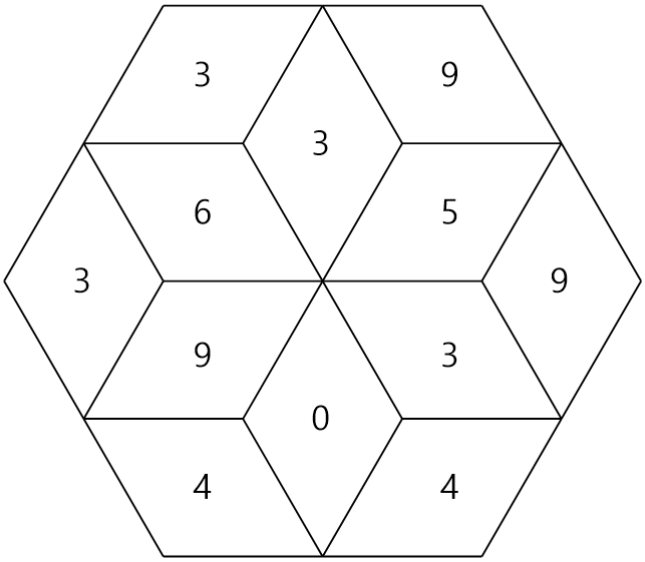
Examples

standard input	standard output
1 2 3 4 5 1 6	9
2 3 14 15 9 2 6 5 3 5 8 97 9 3 2 3 8 4 6 26 4 3 3 8 3 2 7 9 5 0 2	58

Note



The costs of rhombuses given in example 1



The solution for example 2

Problem F. Jenga Game

Time limit:	1 second
Memory limit:	1024 mebibytes

Yesyes and Nono are playing the Jenga game. Jenga has the following rules:

There is a tower consisting of n layers of blocks. Each layer consists of three long blocks. The blocks in each layer lie parallel to each other. The blocks in two neighboring layers are perpendicular to each other. Some blocks might be missing at the start of the game. Two players make their moves alternately. During one move a player must choose a block and remove it from the tower if it remains stable afterward. The tower is stable if all the following conditions are met:

- Each layer contains at least one block.
- If a layer contains exactly one block, it's the middle one.
- The top layer consists of three blocks.

A player who is unable to make any move loses. Given the starting state of the tower, possibly with some blocks already removed, it is always guaranteed that the initial state of the given Jenga tower is stable. Your task is to determine which player will win. Both players always use the best strategy to win against each other. Yesyes plays first. Yesyes and Nono are experts at Jenga, so they don't make any mistakes while removing a block.

Notice that in this version of Jenga, players do not put the blocks on the top of the tower.

Input

The first line of input contains the number of test cases T.

The first line of input for each test case contains the initial height of the Jenga tower N.

Each of the next N lines contains the initial state of each layer as a string of length 3 consisting of `0`s and `1`s, starting from the top layer. `0` means that there is no block at that position and `1` means there is a block at that position. It is always guaranteed that the initial state of the given Jenga tower is stable.

Output

Print the winner in one line for each test case.

Constraints

- $1 \leq T \leq 1\,000$
- $2 \leq N \leq 400\,000$
- The sum of N across all test cases does not exceed $400\,000$.

Example

standard input	standard output
2 6 111 101 010 111 110 111 2 111 101	Yesyes Nono

Problem G. Make RUN Great Again

Time limit:	1 second
Memory limit:	1024 mebibytes

KAIST Clubs Union is a student agency that delegates the university's club funding based on the club's activity records. Taein is a member of RUN — the *marathon* club of KAIST — and has joined the KAIST Clubs Union to make it richer.

There are $N + 1$ clubs at KAIST numbered from 1 to $N + 1$. RUN has a number $N + 1$. KAIST Clubs Union will assign the **scores** to each club, which is a non-negative integer. A club's rank is then defined as (Number of clubs with a higher score than itself) + 1. Note that there can be multiple clubs with the same rank. The funding opportunities for the club will be better if a club's rank is lower.

Taein wishes to fabricate the scores in favor of RUN. Currently, every club **except the RUN** has their scores determined — for each $1 \leq i \leq N$, club i has a score of a_i. Taein can change each club's score under the following conditions:

- Each club's score should not increase after Taein's change.
- Each club's score must be a non-negative integer after Taein's change.
- If Taein reduces the club i's score by T, the KAIST Clubs Union's **skepticism factor** will increase by $T \times b_i$, where b_i is an integer decided for each club based on their influence and information gathering skills.
- The total **skepticism factor** should be **less than** K. If it is at least K, it will trigger an internal investigation, which might cause trouble for Taein.

After the fabrication, Taein will fill in the score for RUN, which will finish the funding delegation process.

To avoid suspicion, Taein wishes to keep the score for RUN as low as possible while keeping the rank small enough for all needed funds. You should find the lowest possible initial score for RUN, such that the final rank of RUN is at most X, and there exists a fabrication scheme that raises the skepticism factor by **less than** K. Since RUN is a club as well, this score has to be a non-negative integer as well.

Input

The first line contains N, K, X, separated by spaces.

The i-th of the next N lines contains a_i and b_i, separated by a space.

Output

Print the lowest possible initial score for RUN, such that the final rank of RUN is at most X, and there exists a fabrication scheme that raises the skepticism factor by **less than** K. Since RUN is a club as well, this score has to be a non-negative integer as well.

Constraints

- $1 \leq N \leq 10^5$
- $1 \leq K \leq 10^{18}$
- $1 \leq X \leq N + 1$
- $0 \leq a_i \leq 10^6$
- $1 \leq b_i \leq 10^6$

Examples

standard input	standard output
4 10 2 100 1 2 100000 50 1 30 8	41
5 15 3 15 4 24 1 10 3 24 2 2 8	10

Problem H. Mosaic

Time limit: 2 seconds
Memory limit: 1024 mebibytes

Mosa is an abstract artist. All of her paintings can be represented as a rectangular grid of black and white unit squares.

Lina is an art collector. His favorite hobby is counting black squares in Mosa's paintings. One day, he took one of her paintings of R rows and C columns, and recorded an R-by-C matrix, such that the number at row r, column c in the matrix equals the number of black unit squares in the 3-by-3 square region centered at the square at row r, column c.

Unfortunately, the very next day Lina found that the artwork is missing! All he has now is the matrix of numbers. Can you help Lina recover Mosa's painting?

Lina might have made a mistake while writing down the numbers, so it is possible that such a painting does not exist for a given matrix.

Input

The first line contains two integers R and C separated by a space.

The next R lines contain C nonnegative integers each, also separated by a space. Each number does not exceed 9.

Output

If such a painting exists, output `1` on the first line, and the painting over the next R lines. Each line should be a string of length C, consisting of characters `B` (for Black) and `W` (for White). If multiple paintings satisfy the condition, output any one of them.

If such a painting does not exist, output `0` on the first line.

Constraints

- $1 \leq R, C \leq 1\,000$

Examples

standard input	standard output
5 5 4 6 6 6 4 6 8 8 8 6 6 8 8 8 6 6 8 8 8 6 4 6 6 6 4	1 BBBBB BBBBB BBWBB BBBBB BBBBB
4 4 0 1 0 1 1 0 1 0 0 1 0 1 1 0 1 0	0

Problem I. Mukjjippa

Time limit:	2 seconds
Memory limit:	1024 mebibytes

Two players A and B are playing a game called **mukjjippa**.

The game consists of several turns.

At the i-th turn ($1 \le i \le n$):

- Each player chooses exactly one from $\{\mathrm{R}, \mathrm{S}, \mathrm{P}\}$ (meaning rock, scissors, and paper, respectively).
- Let X_i and Y_i be the choices of A and B, respectively.
- If $(X_i, Y_i) \in \{(\mathrm{R}, \mathrm{S}), (\mathrm{S}, \mathrm{P}), (\mathrm{P}, \mathrm{R})\}$, then A becomes an attacker for the $(i+1)$-th turn and the game continues.
- Otherwise, if $(X_i, Y_i) \in \{(\mathrm{R}, \mathrm{P}), (\mathrm{S}, \mathrm{R}), (\mathrm{P}, \mathrm{S})\}$, then B becomes an attacker for the $(i+1)$-th turn and the game continues.
- Otherwise, if there is an attacker for the i-th turn, then the attacker becomes a winner and the game ends.
- Otherwise, there is no attacker for the $(i+1)$-th turn and the game continues.

Note that there is no attacker for the first turn.

If the game does not end until the beginning of the $(n+1)$-th turn, nobody is a winner.

The probability distribution of each choice is given. All choices are independent.

Find the probability that A wins.

Input

The first line contains an integer n.

The i-th of the next n lines contains three integers r_i, s_i, and p_i. This means that the probabilities that X_i is R, S, and P are $\frac{r_i}{r_i+s_i+p_i}$, $\frac{s_i}{r_i+s_i+p_i}$, and $\frac{p_i}{r_i+s_i+p_i}$, respectively.

The i-th of the next n lines contains three integers r'_i, s'_i, and p'_i. This means that the probabilities that Y_i is R, S, and P are $\frac{r'_i}{r'_i+s'_i+p'_i}$, $\frac{s'_i}{r'_i+s'_i+p'_i}$, and $\frac{p'_i}{r'_i+s'_i+p'_i}$, respectively.

Output

Let $\frac{x}{y}$ be the probability that A wins, where x and y are coprime integers, and $x \ge 0$ and $y > 0$.

Print the integer z such that $yz \equiv x \pmod{998\,244\,353}$ and $0 \le z < 998\,244\,353$.

It can be proved that such an integer z always exists and is uniquely determined, under the constraints of this problem.

Constraints

- $1 \le n \le 2 \times 10^5$
- $0 \le r_i, s_i, p_i \le 10^6$ $(1 \le i \le n)$
- $r_i + s_i + p_i > 0$ $(1 \le i \le n)$
- $0 \le r'_i, s'_i, p'_i \le 10^6$ $(1 \le i \le n)$
- $r'_i + s'_i + p'_i > 0$ $(1 \le i \le n)$

Examples

standard input	standard output
2 1 0 0 0 1 0 0 1 0 0 1 0	1
2 1 1 1 1 1 1 1 1 1 1 1 1	443664157

Problem J. Running in the Plane

Time limit:	1 second
Memory limit:	1024 mebibytes

You are given a set $S = \{(a_1, b_1), (a_2, b_2), \ldots, (a_n, b_n)\}$ of n points in the plane. All coordinates of S are integers.

A set $T = \{(c_1, d_1), (c_2, d_2), \ldots, (c_m, d_m)\}$ of m 2-dimensional vectors is called a **good set** of S if it satisfies the following:

1. There exists a nonempty finite sequence $((x_0, y_0), (x_1, y_1), \ldots, (x_l, y_l))$ of points in the plane such that
   (a) $(x_0, y_0) = (0, 0)$.
   (b) For all points p in S, there exists an integer i $(0 \le i \le l)$ such that $(x_i, y_i) = p$.
   (c) For all integers i $(0 \le i < l)$, the vector $(x_{i+1} - x_i, y_{i+1} - y_i)$ is in T.
2. For all integers i $(1 \le i \le m)$, two numbers c_i and d_i are integers between -10^{18} and 10^{18} inclusive.

Find any good set of **minimum** size.

Input

The input consists of multiple test cases. The first line contains an integer Q — the number of test cases. The description of the test cases follows. For each test case:

- The first line of the test case contains an integer n — the number of points in S.
- The i-th of the next n lines contains two integers a_i and b_i — the coordinates of each point in S.

Output

For each test case:

- Let $T = \{(c_1, d_1), (c_2, d_2), \ldots, (c_m, d_m)\}$ be a minimum-size good set of S.
- In the first line of the test case, print an integer m — the number of vectors in T.
- In the i-th of the next m lines, print two integers c_i and d_i — the coordinates of each vector.

If there are multiple solutions, print any of them.

It can be proved that, under the constraints of this problem, a good set of S with size at most $10 \times n$ always exists.

Constraints

- $1 \le Q \le 50\,000$
- The sum of n over all test cases does not exceed 10^5.
- $2 \le n \le 10^5$
- $-10^8 \le a_i, b_i \le 10^8$ $(1 \le i \le n)$
- $(a_i, b_i) \ne (a_j, b_j)$ $(1 \le i < j \le n)$
- $m \ge 0$
- $-10^{18} \le c_i, d_i \le 10^{18}$ $(1 \le i \le m)$
- $(c_i, d_i) \ne (c_j, d_j)$ $(1 \le i < j \le m)$

Example

standard input	standard output
2 2 -30 30 -50 50 3 2 1 1 0 4 1	1 -10 10 2 1 0 1 1

Note

In the first test case, $T = \{(-10, 10)\}$ is a minimum-size good set of $S = \{(-30, 30), (-50, 50)\}$.

We can take a sequence $((0, 0), (-10, 10), (-20, 20), \underline{(-30, 30)}, (-40, 40), \underline{(-50, 50)})$. Here, the underlined points are in S.

In the second test case, $T = \{(1, 0), (1, 1)\}$ is a minimum-size good set of $S = \{(2, 1), (1, 0), (4, 1)\}$.

We can take a sequence $((0, 0), \underline{(1, 0)}, \underline{(2, 1)}, (3, 1), \underline{(4, 1)})$.

Problem K. Same Segment

Time limit:	1 second
Memory limit:	1024 mebibytes

You have a sequence a of N integers between 0 and K inclusive. M segments are given, where ith segment is $[l_i, r_i]$. We want $\sum_{j=l_i}^{r_i} a_j = K$ to satisfy for every segment. Determine whether there exists such sequence a.

Input

Each test data contains one or more test cases. The first line contains an integer T — the number of test cases for this input file.

First line of each test case contains 3 integers N, M, K.

i-th of the next M lines contain two integers l_i and r_i: left and right end of i-th segment.

Output

For each test case, if a sequence that satisfies the condition exists, output the elements of the sequence. In case of multiple answers you may output any of them.

If no valid sequence exists, output a single integer -1.

Constraints

- $1 \le T \le 10^5$
- $2 \le N \le 4 \times 10^5$
- $1 \le M \le \min\left(2 \times 10^5, \frac{N(N+1)}{2}\right)$
- $1 \le K \le 20$
- $1 \le l_i \le r_i \le N$
- $(l_i, r_i) \ne (l_j, r_j)$ for $i \ne j$
- Sum of N over every test case does not exceed 4×10^5.
- Sum of M over every test case does not exceed 2×10^5.

Example

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

Problem L. Simple Tree Decomposition Problem

Time limit: 3 seconds
Memory limit: 1024 mebibytes

BOOM! Dohoon's head has exploded while solving the tree decomposition practice problem given as an assignment while attending the Summer School on Combinatorics and Algorithms at KAIST. Dohoon's brain, now unable to focus on the problem, is idling away time performing 'decomposition' on a 'tree' instead of doing tree decomposition on general graphs.

Specifically, Dohoon is given a tree with N vertices. He plans to decompose the tree into a collection of connected components as follows:

1. Dohoon will select zero or more edges from the tree and remove them from the tree. Let S be the set of removed edges in this procedure.
2. After the edges in S are removed, each connected component in the resulting graph must have either A or B vertices.

Help Dohoon find the number of different ways to decompose the tree as given above. To be specific, determine the number of possible sets of edges S that satisfy the given conditions.

Note that a tree is a connected, acyclic, undirected graph, where each undirected edge is an unordered pair of vertices.

Input

The first line contains three space-separated integers, N, A, B.

The i-th of the following $N-1$ lines contains two space-separated integers x_i and y_i, denoting that the i-th edge connects vertices x_i and y_i in the tree.

Output

Print the number of possible sets S that satisfy the conditions given in the problem, modulo 10^9+7.

Constraints

- $1 \le N \le 100\,000$
- $1 \le A < B \le 500$
- $1 \le x_i < y_i \le N$ $(1 \le i \le N-1)$
- It is guaranteed that the given edges form a tree.

Example

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

Problem M. White-Black-Tree

Time limit: 2 seconds
Memory limit: 1024 mebibytes

There is a tree with N vertices. Vertices are numbered from 1 to N.

Each vertex is colored either white or black. You can swap the colors of two adjacent vertices any number of times you want. A **final tree** is any tree formed by completing a finite number of swaps in the original tree. The **swap cost** of a final tree is equal to the number of swaps you have made.

A white tree is the connected subgraph of the final tree which connects all the white vertices of the final tree using the minimum number of edges. Likewise, a black tree is the connected subgraph of the final tree which connects all the black vertices of the final tree using the minimum number of edges.

The **edge cost** of a final tree is equal to the sum of the number of edges in the white tree and the number of edges in the black tree. Note that the edge cost is calculated only after all swaps have been completed.

The total cost of a final tree is the sum of its swap cost and edge cost.

Find the minimum possible total cost of a final tree.

Input

The first line of input contains the number of vertices N.

The second line of input contains a string s of length N consisting only of `W` and `B`. If the i-th character of s is `W`, vertex i is white, and if the i-th character is `B`, vertex i is black. It is guaranteed that there are at least one `W` and at least one `B` in the string s.

Each of the following $N-1$ lines contains u and v denoting that there is an edge connecting vertex u and vertex v.

It is guaranteed that the input forms a valid tree.

Output

The first line of output should contain the minimum possible total cost of a final tree.

Constraints

- $2 \leq N \leq 5 \times 10^5$
- $1 \leq u < v \leq N$

Examples

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