Problem A. Add, Remove, Transform

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	1024 mebibytes

You are given a tree with n vertices. You can repeat the following operation **at most** 10^5 times.

- Choose four distinct vertices $1 \le v_1, v_2, v_3, v_4 \le n$ such that there exist edges between v_1 and v_2, v_2 and v_3, v_3 and v_4. Remove these edges and add edges between v_1 and v_3, v_1 and v_4, v_2 and v_4.

Your task is to transform the given tree so that its *diameter* is **at most** 3. Find a sequence of operations that does so.

Input

The first line of the input contains one integer n ($4 \le n \le 100$).

The i-th of the following $n-1$ lines contains two integers x_i and y_i ($1 \le x_i, y_i \le n$; $x_i \ne y_i$ for $1 \le i \le n-1$), meaning that the i-th edge connects vertices x_i and y_i in the tree.

You may assume that the given edges form a tree.

Output

At the first line, print k, the number of operations ($0 \le k \le 10^5$).

In each of the next k lines, print four integers v_1, v_2, v_3, v_4 separated by spaces. The values v_1, v_2, v_3, and v_4 should satisfy the conditions above.

If there are multiple solutions, print any one of them. It can be proven that there exists at least one way to achieve the goal.

Note that you do not have to minimize k.

Example

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

Note

The *distance* between two vertices u and v is defined as the number of edges on the unique simple path from u to v.

The *diameter* of a tree is the maximum *distance* between any two vertices.

Problem B. Big Sieve Game

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

After mastering the sieve of Eratosthenes, Alice excitedly created a puzzle game that made use of it. The rules of the puzzle game are as follows:

- An array $p_1, p_2, \ldots, p_n$ is given. Initially, all p_i are 0.
- A target array $t_1, t_2, \ldots, t_n$ is given.
- The player can perform the following two types of operations:
  - Choose i and increase p_j by 1 for every j divisible by i.
  - Choose i and decrease p_j by 1 for every j divisible by i.
- The puzzle is solved when $p = t$ after applying zero or more operations.

Alice aims to solve the puzzle using the fewest operations, showcasing her puzzle-solving skills. Please help Alice find the minimum number of operations to solve the puzzle.

Input

The first line contains a single integer n ($1 \le n \le 2 \cdot 10^5$).

The second line contains n space-separated integers t_i: the elements of the array t ($-10^9 \le t_i \le 10^9$).

Output

Print the minimum number of operations to solve the puzzle. If the puzzle is unsolvable, print `-1`.

Examples

standard input	*standard output*
4 1 1 1 0	2
7 0 1 1 1 0 2 -1	3

Problem C. Catch The Flea

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

You have placed glues on each cell of an $n \times m$ grid to create a rectangular flea trap. Each glue has a *weak direction*; if a flea on the glue jumps towards its *weak direction*, the flea can jump out of the glue.

More precisely, each glue is represented by `U`, `D`, `L`, or `R`, meaning up, down, left, and right respectively.

In a single jump, a flea can move any distance up to k cells in the weak direction. If a flea jumps out of the rectangle, we say that the flea has *escaped*.

You became curious about how efficient your trap is. If a flea that is placed on a cell of the trap can escape after consecutive jumps, we call the cell an *escapable cell*. Your task is to count the number of escapable cells.

Input

The first line of input contains three integers: the trap sizes n and m and the jump limit k ($1 \le n, m, k \le 2000$).

Each of the next n lines contains a string of length m consisting of letters `U`, `D`, `L`, and `R`. These lines indicate the *weak direction* of each glue.

Output

Print the number of escapable cells.

Example

standard input	*standard output*
5 5 2 DDDRD DDDDD RDLUL UURUU UUUUU	14

Note

Fleas which start from $(1,3)$, $(1,4)$, $(1,5)$, $(2,3)$, $(2,4)$, $(2,5)$, $(3,4)$, $(3,5)$, $(4,3)$, $(4,4)$, $(4,5)$, $(5,3)$, $(5,4)$, and $(5,5)$ can escape. For example:

- A flea at $(1,3)$ can escape by jumping along the following path:
  $(1,3) \to (2,3) \to (4,3) \to (4,4) \to (3,4) \to (1,4) \to$ out of the trap
- A flea at $(1,4)$ can escape by jumping two cells to the right.

Problem D. Decorative Birds

Input file:	*standard input*
Output file:	*standard output*
Time limit:	6 seconds
Memory limit:	1024 mebibytes

There are n adorable geese in KAIST campus. To celebrate these lovely creatures, KAIST will host "KAIST Geese Show" for students. The main content of the show is simply feeding the geese and enjoying their delightful reactions.

You have the honor of being chosen as the representative feeder for the students. Your mission is to make the show as cute as possible by feeding the geese optimally. The i-th goose approaches you at time T_i and will eagerly wait for food for a duration of L. More precisely, the i-th goose is available to eat food during the time interval $T_i \le x \le T_i + L$. After time $T_i + L$, the goose will lose interest and leave.

The i-th goose has speed A_i and cuteness C_i. No two geese have the same speed. If you throw a piece of food to the waiting geese at any time, the fastest goose among those will take it. Then the goose will proudly display its cuteness for all to see, adding its cuteness value to the overall cuteness score of the show. Note that there are some noisy geese, so C_i can be negative. After consuming a piece of food, the goose will be satisfied and will leave.

You have the freedom to throw as many pieces of food as you desire, with no constraints on frequency or quantity. Your goal is to determine the maximum cuteness score achievable for the show.

Input

The first line of input contains two integers, n and L ($1 \le n \le 3 \cdot 10^5$; $1 \le L \le 10^9$).

Each of the following n lines contains three space-separated integers: the i-th line contains A_i, C_i, and T_i ($1 \le A_i \le N$; $-10^9 \le C_i \le 10^9$; $0 \le T_i \le 10^9$).

Output

Output the maximum cuteness score of the show.

Example

standard input	*standard output*
6 5 6 -1 7 4 -5 9 1 3 11 5 -4 13 2 4 14 3 6 7	9

Problem E. Excellent HLD

Input file:	*standard input*
Output file:	*standard output*
Time limit:	4 seconds
Memory limit:	256 mebibytes

You are given a rooted tree consisting of n vertices. Its root is vertex 1. Let us consider a *heavy-light decomposition* of the tree, where each edge is either a *heavy edge* or a *light edge*. For each vertex, among all edges connecting the vertex with its children, at most one edge can be a heavy edge.

In this problem, we have a multiset of simple paths T, which is initially empty. We will assign each edge to be a heavy edge or a light edge according to T, satisfying the condition above.

Each time an update is done on T, your task is to find an assignment of edges that minimizes the sum of the number of light edges on every path in T.

There are q updates given. Each update consists of three integers: s, e, and k. They mean that k copies of the simple path from s to e are inserted into T. After each update, find the minimum sum of the number of light edges on every path in T.

Input

The first line of input contains two space-separated integers n and q ($2 \le n \le 10^5$; $1 \le q \le 10^5$).

The i-th of the following $n - 1$ lines contains two space-separated integers x_i and y_i, meaning that the i-th edge connects vertices x_i and y_i in the tree ($1 \le x_i, y_i \le n$; $x_i \ne y_i$). It is guaranteed that the given edges form a tree.

The i-th of the following q lines contains three space-separated integers, s, e, and k, describing each update ($1 \le s, e \le n$; $s \ne e$; $1 \le k \le 10^9$).

The updates are processed in the input order. The updates are permanent: the changes made in each update persist in further updates as well.

Output

For each of the q updates, print a line with the answer after this update.

Examples

standard input	*standard output*
3 3 1 2 3 1 1 3 2 1 3 3 1 2 2	0 0 2
5 5 3 4 2 4 1 2 5 3 5 4 2 1 2 4 3 4 1 5 3 4 1 2 2	0 0 0 0 0
8 8 4 6 8 4 1 6 5 1 2 1 3 2 7 3 2 7 1 8 2 1 5 3 6 8 3 5 1 4 2 6 7 1 5 6 4 6 2 3	0 1 7 12 14 15 23 26

Problem F. Full Irreducibility

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

A permutation is called *irreducible* if none of its prefixes form a permutation, except the full permutation itself. For example, $[2, 3, 1]$ and $[4, 1, 2, 3]$ are irreducible, while $[2, 1, 3]$ and $[1, 3, 2]$ are not.

You are given a permutation p of length n. In one operation, you can choose any two adjacent positions and swap the values at these positions.

Find the minimum number of operations to transform p into an irreducible permutation, and the corresponding sequence of operations itself.

Input

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

The first line of each test case contains a single integer n ($1 \le n \le 10^5$).

The second line of each test case contains n integers $p_1, \ldots, p_n$ ($1 \le p_i \le n$).

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

Output

For each test case, print two lines:

On the first line, print an integer k: the minimum number of operations that make p irreducible.

On the second line, print k space-separated integers $s_1, \ldots, s_k$ ($1 \le s_i \le n - 1$) where s_i and $s_i + 1$ are the positions of the values you intend to swap. The operations are performed sequentially in the order specified by your output. If there are multiple solutions, print any one of them.

Example

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

Problem G. Good Triangle

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

You are given n distinct points on a two-dimensional plane.

We define the distance between two points $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ as $d(P, Q) = |x_1 - x_2| + |y_1 - y_2|$.

Let us say that three distinct points U, V, W form a *good triangle* if there exists a point T such that $d(U, T) = d(V, T) = d(W, T)$. Note that T does not have to be a lattice point.

Find the number of good triangles that can be formed by the given points.

Input

The first line of the input contains one integer N ($3 \le N \le 5 \cdot 10^5$).

The i-th of the next N lines contains two space-separated integers x_i and y_i, meaning that the coordinate of the i-th point is (x_i, y_i) ($-10^9 \le x_i, y_i \le 10^9$; $(x_i, y_i) \ne (x_j, y_j)$ if $i \ne j$).

Output

Print one integer: the number of good triangles that can be formed by the given points.

Examples

standard input	*standard output*
5 1 -1 1 5 5 7 1 3 4 2	9
10 -2 -1 -2 2 -1 -2 -1 -1 -1 1 0 1 1 -1 1 2 2 -1 2 1	108

Problem H. Holes in Queue

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

A queue is a linear data structure that can either

1. insert an element at the back (push), or
2. delete an element at the front (pop).

However, we misimplemented the pop function. For each pop query, instead of deleting only one element at the front, we simultaneously delete n certain elements from different indices.

Specifically, there are n distinct locations where our pop function deletes its elements: $a_1, a_2, \ldots, a_n$. The queue is indexed from 1 starting at the front, and after each pop query, the remaining elements are renumbered from 1 again.

We are curious what the misimplemented queue will look like after d pop operations.

In order to conduct an experiment, we first pushed infinitely many numbers in the queue starting from 1. So, the initial queue will look like "1 2 3 4 5 6 7 8...".

Then, without further push operations, we will pop the queue d times.

For example, assume the current queue is "1 2 3 4 5 6 7 8...". If we delete the 2nd and 5th element, the queue will change to "1 3 4 6 7 8 9 10...". If we do it again, the queue will become "1 4 6 8 9 10 11 12...". And so on.

We want to process q queries. Each query consists of a single integer x, which means that we need to calculate the number at the x-th position in the queue after d pop operations.

Input

The first line of input contains an integer n ($1 \le n \le 5 \cdot 10^5$).

The second line contains n space-separated integers $a_1, a_2, \ldots, a_n$ denoting the locations we delete for each pop operation ($1 \le a_i \le 10^{12}$; all a_i are distinct).

The third line contains two space-separated integers, q and d, denoting the number of queries and the number of pop operations, respectively ($1 \le q \le 5 \cdot 10^5$; $1 \le d \le 10^{12}$).

Each of the following q lines contains an integer x denoting a query ($1 \le x \le 10^{12}$).

Output

Output q lines, where the i-th line contains a single integer denoting the answer to the i-th query.

Examples

standard input	*standard output*
2 2 5 8 2 1 2 3 4 5 6 7 8	1 4 6 8 9 10 11 12
3 7 1 32 8 5 2 8 7 17 26 19 3 1	8 18 17 27 39 29 10 6

Problem I. Isn't It Beautiful?

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

We define the beauty of an array as its minimum excluded value (MEX): the smallest non-negative integer that does not belong to the array. A larger MEX corresponds to a more beautiful array.

You are given an array a of length n. You want to enhance it in terms of its beauty. To achieve this, you can choose a non-negative integer x and replace each element a_i with a_i & x. Here, & denotes the bitwise AND operator: each bit of the result is equal to the logical AND of the respective bits of the operands.

Find the value of x that maximizes the beauty of the array.

Input

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

Each test case is given on two lines.

The first of these lines contains a single integer n ($1 \le n \le 10^5$).

The second line contains n integers $a_1, \ldots, a_n$ ($0 \le a_i < 2^{30}$).

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

Output

For each test case, print a single integer x that maximizes the MEX of the array formed by taking the bitwise AND of each element of a with x. This value should satisfy $0 \le x < 2^{30}$; it can be shown that there always exists an optimal x in that range. If there are multiple solutions, print any one of them.

Example

standard input	*standard output*
1 6 13 11 40 10 33 19	23

Note

In the sample, we can choose $x = 23$, so the new array will be

$$[13\&23,\ 11\&23,\ 40\&23, 10\&23,\ 33\&23,\ 19\&23],$$

which is $[5, 3, 0, 2, 1, 19]$ with a MEX of 4. Another possible answer is $x = 19$.

Problem J. Joy Of Sleep

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	1024 mebibytes

There are n chameleons residing in the *Chameleon Village* which can be viewed as a two-dimensional plane. The i-th chameleon is located at (x_i, y_i) and has color c_i.

Deep into the night, only the first chameleon is awake, while all the other chameleons are asleep.

Suddenly, an urgent issue has arisen for the n-th chameleon, thus it needs to be awakened. Physical contact is required to wake up a chameleon. Therefore, to wake up a chameleon, other chameleon must either walk to its location, or extend its tongue to touch it.

Chameleons can move in eight directions: up, down, left, right, or diagonally. Specifically, a chameleon at (x, y) can move to one of the following locations in one second: $(x-1, y-1)$, $(x-1, y)$, $(x-1, y+1)$, $(x, y-1)$, (x, y), $(x, y+1)$, $(x+1, y-1)$, $(x+1, y)$, or $(x+1, y+1)$.

Additionally, a chameleon can extend its tongue vertically or horizontally. In other words, a chameleon at (x, y) can extend its tongue to reach locations $(x+c, y)$ or $(x, y+c)$ for any (positive or negative) integer c. Extending the tongue takes no time. However, chameleons of the same color can barely see each other, making it challenging for them to aim, so they cannot extend the tongue towards each other.

When a tongue is extended, it doesn't matter if there are other chameleons along the path; only the chameleon located at the destination of the tongue is awakened, and other chameleons along the path are not disturbed. A chameleon may also walk through another chameleon's position without waking them up.

Since chameleons are sleepy, they immediately fall asleep at their current location after waking up another chameleon.

Determine the minimum number of seconds required to wake up the n-th chameleon.

Input

The first line of input contains two integers: the number of chameleons n and the number of different colors m ($2 \le n \le 10^5$; $1 \le m \le n$).

The i-th of the next n lines contains three integers: the initial coordinates x_i and y_i and the color c_i of the i-th chameleon ($-10^9 \le x_i, y_i \le 10^9$; $1 \le c_i \le m$). No two chameleons start at the same point. There is a chameleon of each of the m colors.

Output

Output the minimum time (in seconds) required to wake up the n-th chameleon.

Example

standard input	*standard output*
7 3 -5 0 1 -3 3 2 6 10000 2 5 3 3 3 -7 3 0 3 2 7 0 1	4

Note

The following is an explanation for the example. In the following way, it is possible to wake up the 7th chameleon in 4 seconds.

- The 1st chameleon wakes up the 2nd chameleon by first walking from $(-5, 0)$ to $(-3, 0)$ in 2 seconds, then extending its tongue to $(-3, 3)$.
- The 2nd chameleon wakes up the 4th chameleon by extending its tongue from $(-3, 3)$ to $(5, 3)$. The 6th chameleon is sleeping in the middle of the path, but it does not get disturbed.
- The 4th chameleon wakes up the 3rd chameleon by first walking from $(5, 3)$ to $(6, 4)$ in 1 second, then extending its tongue to $(6, 10000)$.
- The 3rd chameleon wakes up the 7th chameleon by walking from $(6, 10000)$ to $(7, 9999)$ in 1 second, then extending its tongue to $(7, 0)$.

Problem K. Kids And Sequence Game

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	1024 mebibytes

Alice and Bob are playing the *Binary Game*. Initially, there is a sequence $a_1, a_2, \ldots, a_n$ of n positive integers. The players take turns starting with Alice. On each turn, the player performs the following action:

- Choose an integer i ($1 \le i \le n$) such that $a_i > 0$. Then get ($a_i \bmod 2$) points and apply $a_i := \left\lfloor \frac{a_i}{2} \right\rfloor$.

The game ends when all integers in the sequence become zeroes. The *result* of the game is the value $A - B$ where A is Alice's points and B is Bob's points. Alice's goal is to maximize the result, and Bob's goal is to minimize it. What will be the result of the game if both players play optimally?

Input

The first line of input contains a single integer n ($1 \le n \le 5 \cdot 10^4$).

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i < 2^{63}$).

Output

Output a single integer: the result of the game (Alice's points minus Bob's points) if both players play optimally.

Example

standard input	*standard output*
5 13 29 10 1 26	3

Problem L. Lottery

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

You are a big fan of Korea's biggest rockstar, Koosaga. Excitingly, Koosaga has announced a lottery event offering fans a once-in-a-lifetime opportunity for a one-on-one meeting.

Koosaga sells n kinds of albums. When you purchase the i-th kind of album, it costs you a_i won, and you get b_i lottery tickets. You can purchase multiple copies of the same album if you wish.

On the day of the draw, a large roulette wheel containing cells with participant names will decide the winner. Every cell on the wheel has an equal chance of being selected. The number of cells with your name corresponds to the number of lottery tickets you've amassed.

Koosaga will give the wheel a single spin to pick the winner. But if luck isn't on your side initially, there's still hope! By paying r won, you can request Koosaga to spin the wheel again. You can pay for as many re-spins as you desire.

Armed with insider knowledge, you've learned that the cumulative number of cells attributed to other participants is s. Importantly, none of them will opt for a re-spin.

Your challenge is to find an optimal strategy to win with the minimum expected cost.

Input

The first line contains three integers: n, s, and r ($1 \le n \le 10^5$; $1 \le s \le 10^6$; $1 \le r \le 10^6$). The i-th of the next n lines contains two integers: a_i and b_i ($1 \le a_i \le 300$; $1 \le b_i \le 5000$).

Output

Print a single line with two positive integers x and y which must be coprime. The value $\frac{x}{y}$ must be the minimum expected cost to win. It is guaranteed that the minimum expected cost can be expressed in this format.

Example

standard input	*standard output*
3 11 3 1 3 2 7 5 13	63 10

Problem M. Make It Regular

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

You are given a bracket sequence consisting of n opening brackets and n closing brackets. Let S be a **non-empty** set of integers between 1 and $2n$, inclusive. You can choose two indices in S, not necessarily adjacent, and swap the brackets of the bracket sequence at those two positions.

Find the number of S that make it possible to obtain a regular bracket sequence by repeatedly applying this operation an arbitrary number of times. As this number may be very large, find it modulo the prime number 998 244 353.

Input

The first line contains one integer n ($1 \le n \le 3000$).

The second line contains a string of $2n$ brackets, either "(" or ")". The given bracket sequence contains n opening brackets and n closing brackets.

Output

Print the number of all possible S modulo 998 244 353.

Examples

standard input	*standard output*
3 ())(()	36
6 ()))(())()((	1536