Problem A. Almost Correct

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Given a 01-sequence (sequence consisting only of 0s and 1s) s of length n, you need to find two arrays x and y such that

- Array x and y are of equal length, and the length should not exceed 120.
- $1 \le x_i < y_i \le n$ for all $1 \le i \le m$, where m is the length of the two arrays.
- Algorithm SORT(x, y) can sort all 01-sequences of length n correctly **except for** the given one. Here algorithm SORT(x, y) takes as input a 01-sequence a of length n, and is described as follows:
  - Iterate over all integers from 1 to m. For each integer i during the iteration, swap a_{x_i} and a_{y_i} if $a_{x_i} > a_{y_i}$.

Input

The input contains multiple test cases.

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

For each test case, the first line contains an integer n ($2 \le n \le 16$).

The next line contains s, the 01-sequence of length n. It is guaranteed that s is not sorted, i.e., there exists some integers $1 \le i < j \le n$ such that $s_i = 1$ and $s_j = 0$.

Output

For each test case, output m ($0 \le m \le 120$) in the first line, indicating the length of array x and y.

In the i-th of the next m lines, print x_i and y_i ($1 \le x_i < y_i \le n$) separated by a space.

Example

standard input	standard output
2 2 10 3 110	0 2 1 2 2 3

Problem B. Anticomplementary Triangle

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

There are n points $P_1, P_2, \ldots, P_n$ in a plane forming a convex polygon. The coordinates of the i-th point P_i are (x_i, y_i). You need to find three pairwise distinct indices $r, s, t \in \{1, 2, \ldots, n\}$, such that there exists a triangle $\triangle ABC$ satisfying

- $\triangle ABC$ is the anticomplementary triangle of $\triangle P_r P_s P_t$.
- P_i is inside or on the edge of $\triangle ABC$ for all $1 \le i \le n$.

$\triangle ABC$ is the anticomplementary triangle of $\triangle XYZ$ if X, Y, Z are the midpoints of edge BC, CA, AB respectively.

Input

The first line contains an integer n ($3 \le n \le 10^6$), indicating the number of points.

The i-th of the next n lines contains two integers x_i and y_i ($|x_i|, |y_i| \le 10^9$), indicating the coordinates of P_i. The vertices are guaranteed to form a convex polygon, and are given in counterclockwise order.

Output

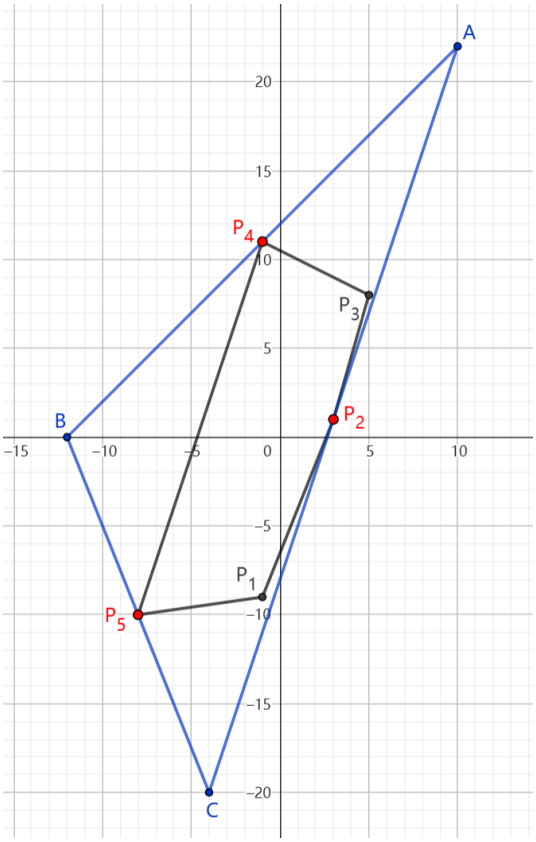
Output three integers r, s and t indicating the selected indices. If there are multiple solutions, print any of them. You can output r, s, t in an arbitrary order.

Examples

standard input	standard output
4 0 0 1 0 1 1 0 1	1 2 3
5 -1 -9 3 1 5 8 -1 11 -8 -10	2 4 5

Note

Here is the illustration for the second example.


A
B
C
P1
P2
P3
P4
P5
20
15
10
5
−5
−10
−15
−20
−15
−10
−5
0
5
10

Problem C. Carrot Trees

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

Walk Alone planted n magical carrot trees in a row. Let a_i denote the number of carrots on the i-th tree (note that a_i is a **real number**). Initially, there are no carrots on any tree. On each day, he can do one of the following two operations on the trees:

- `1 l r x`. Water all carrot trees between ℓ and r inclusively. After that, a_i will increase by x/k immediately for all $\ell \le i \le r$, where k is a constant in all operations.
- `2 l r`. Pick a carrot from all trees between ℓ and r inclusively and having at least one carrot. After that, a_i will decrease by 1 for all integers i where $\ell \le i \le r$ and $a_i \ge 1$.

After m days, Walk Alone has picked many carrots, and he wants you to tell him the total number of carrots he harvested.

Input

The first line contains three integers n ($1 \le n \le 10^6$), m ($1 \le m \le 2 \cdot 10^5$) and k ($1 \le k \le 10^9$), indicating the number of carrots, the number of operations, and the constant in the first operation.

Each of the following m lines indicates an operation, the format of which is as above. It is guaranteed that $1 \le \ell \le r \le n$ and $1 \le x \le 10^9$.

Output

Output an integer indicating the total number of carrots.

Example

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

Problem D. Chocolate

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Walk Alone has $n \times m$ pieces of chocolates placed in a board with n rows and m columns. The coordinates of the i-th row and the j-th column is (i, j).

Walk Alone wants to play a game with Kelin. They will eat the chocolates in turn with Kelin moving first. In each turn, a player can choose two integers i, j $(1 \le i \le n, 1 \le j \le m)$, and then eat all chocolates with coordinates (x, y) where $x \le i$ and $y \le j$. One must eat at least one piece of chocolate in his turn, and the one who eats the last piece loses the game.

Suppose that both Walk Alone and Kelin are intelligent enough and will play optimally, you should decide who will win the game. If Walk Alone will win, print 'Walk Alone'; otherwise, print 'Kelin' (without quotes).

Input

The only line of the input contains two integers n and m $(1 \le n, m \le 10^9)$, indicating the size of the board.

Output

Print 'Walk Alone' if he wins, and 'Kelin' otherwise (without quotes).

Examples

standard input	standard output
2 3	Kelin
1 1	Walk Alone

Problem E. Heap

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

You are given a tree with n nodes rooted at node 1, and the i-th node has a value a_i. You need to assign a new value b_i to each node, such that $b_{p_i} \leq b_i$ for each node i and its parent p_i. Output the minimum value of $\sum_{i=1}^{n}(a_i - b_i)^2$.

Input

The first line contains an integer n ($1 \leq n \leq 2 \cdot 10^5$) indicating the number of nodes in the tree.

The second line contains n integers, and the i-th integer a_i ($0 \leq a_i \leq 10^4$) indicates the original value of the i-th node.

Each of the next $(n-1)$ lines contains two integers u and v, indicating an edge between u and v.

Output

Output a real number indicating the minimum value of $\sum_{i=1}^{n}(a_i - b_i)^2$. Your answer will be considered correct if and only if the absolute or relative error of your answer to the correct answer is less than or equal to 10^{-6}.

Examples

standard input	standard output
3 1 0 2 1 2 1 3	0.500000000
5 9 5 2 7 3 1 2 1 3 2 4 2 5	28.750000000
6 1 2 3 4 5 6 1 2 2 3 3 6 1 4 1 5	0.000000000

Problem F. Intersection

Input file:	standard input
Output file:	standard output
Time limit:	3 seconds
Memory limit:	256 megabytes

Given n circles in a plane, you need to draw a circle, and find the maximum number of given circles it can intersect with.

Two circles intersect with each other if and only if they share at least one common point. It is guaranteed that any two given circles do not share a common point and do not contain each other. The circle you draw may degenerate into a line (a circle with infinite radius) or a point.

Input

The first line contains an integer n ($1 \le n \le 150$), indicating the number of circles.

Each of the next n lines contains three integers x_i, y_i ($|x_i|, |y_i| \le 10^3$) and r_i ($1 \le r_i \le 10^3$), indicating the coordinates and the radius of a circle.

Output

Output an integer indicating the maximum number of intersected circles.

Examples

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

Problem G. LCRS Transform

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 seconds
Memory limit:	256 megabytes

Walk Alone has a binary tree T with n nodes labelled from 1 to n rooted at node 1. Each node u in the tree has at most two children, the left child $\ell_{T,u}$ and the right child $r_{T,u}$, where $\ell_{T,u}$ denotes the label of the left child and $\ell_{T,u} = 0$ if u does not have a left child, and $r_{T,u}$ similarly. We omit the subscript $_T$ if T is clear from the context. The right child of u exists only if u has a left child.

He once performed an operation named left-child right-sibling (LCRS) transform on the tree for exactly k times. In each operation, he would

- Visit all vertices in descending order of depth. For each vertex u with two children ℓ_u and r_u, remove edge (u, r_u) and let the right child of ℓ_u be r_u. It can be shown that ℓ_u does not have a right child at this time.
- For each vertex u with a right child and no left child, i.e., $r_u \neq 0$ and $\ell_u = 0$, swap ℓ_u and r_u.

You can refer to the following code for better understanding.

```
void dfs(int u) {
    // l[u]/r[u]: The left/right child of node u
    // Recursively visit all children of node u
    if (l[u] != 0) dfs(l[u]);
    if (r[u] != 0) dfs(r[u]);

    if (r[u] != 0) {
        // Here l[u] != 0 because r[u] != 0
        if (l[l[u]] == 0) l[l[u]] = r[u];
        else r[l[u]] = r[u];
        r[u] = 0;
    }
}
void LCRS() {
    dfs(1);
}
```

However, he forgot what the original tree is like. Now he wants you to tell him the number of different possible initial trees modulo 998 244 353. Two trees T_1 and T_2 are considered different if and only if there exists a node u such that $(\ell_{T_1,u}, r_{T_1,u}) \neq (\ell_{T_2,u}, r_{T_2,u})$.

Input

The first line contains two integers n ($1 \leq n \leq 10^5$) and k ($0 \leq k \leq 10^5$), the number of nodes and the number of operations Walk Alone has done, respectively.

The i-th of the next n lines contains two integers ℓ_i and r_i ($0 \leq \ell_i, r_i \leq n$), indicating that the left and right child of i in the resulting tree are ℓ_i and r_i respectively. If i has no left (right) child, then $\ell_i = 0$ ($r_i = 0$). It is guaranteed that the given graph is a tree rooted at node 1, and that $r_i \neq 0$ only if $\ell_i \neq 0$.

Output

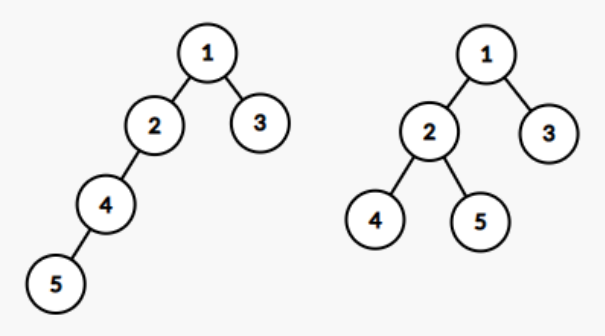
Output an integer indicating the number of possible initial trees modulo 998 244 353.

Examples

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

Note

In the first example, the two possible trees are illustrated as below.



Problem H. Matches

Input file:	standard input
Output file:	standard output
Time limit:	3 seconds
Memory limit:	256 megabytes

Walk Alone has two rows of matches, each of which contains n matches. He defines the distance between them as $d = \sum_{i=1}^{n} |a_i - b_i|$, where a_i and b_i represent the height of the i-th match in the first row and the second row, respectively. Walk Alone wants to beautify the matches by shortening the distance, and you are asked to find out the minimum distance after performing **at most one swap within one row**. Note that the height can be negative.

Input

The first line contains one integer n ($1 \le n \le 10^6$), denoting the number of matches in each row.

The second and the third line both contains n integers, representing the height of matches in the first row and the second row, respectively. All of these numbers satisfy $|a_i|, |b_i| \le 10^9$.

Output

Print one line containing a single integer, the minimum distance you can get.

Examples

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

Note

In the second example, one of the best strategies is to swap a_1 and a_4, and the minimum distance is $|9 - 7| + |3 - 3| + |9 - 7| + |4 - 3| = 5$.

Problem I. Random

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

Walk Alone has a strange random number generator, the pseudocode of which is as follows:

```
uintk rand(uintk x, int A[]) {
    for (int i = 0; i < n; ++i) {
        if (A[i] >= 0) x ^= x << A[i];
        else x ^= x >> -A[i];
    }
    return x;
}
```

Here '`uintk`' denotes the type of unsigned integers with k binary bits, and '`n`' denotes the size of array A. You can use '`std::bitset<k>`' in C++ to implement all operations involving '`uintk`'.

Now Walk Alone is given an array A and an integer k. He found that for any integer x ($0 \le x < 2^k$), there exists a positive integer T such that $\text{rand}_T(x, A) = x$, where

$$\text{rand}_T(x, A) = \begin{cases} \text{rand}(\text{rand}_{T-1}(x, A), A), & T \ge 1 \\ x, & T = 0 \end{cases}$$

Let $f(x) = \min\{T > 0 \mid \text{rand}_T(x, A) = x\}$. Walk Alone wants to know the expected value of $f(x)$ modulo 998 244 353 if x is chosen from $\{0, 1, \ldots, 2^k - 1\}$ with equal probability.

It can be shown that the answer can always be expressed as an irreducible fraction x/y, where x and y are integers and $y \not\equiv 0 \pmod{998\,244\,353}$. Output such an integer a that $0 \le a < 998\,244\,353$ and $a \cdot y \equiv x \pmod{998\,244\,353}$.

Input

The first line contains two integers n ($0 \le n \le 10^5$) and k ($1 \le k \le 32$), the size of array A and the length of type '`uintk`' respectively.

The second line contains n integers, the i-th of which is A_i ($0 < |A_i| < k$).

Output

Print the expected number of $f(x)$ modulo 998 244 353 in one line.

Examples

standard input	standard output
1 2 1	499122178
0 32	1
3 5 2 3 -4	623902730

Note

In the first example, when $A = [1]$, the values of function rand and f are as follows:

- $\text{rand}(0, A) = 0, f(0) = 1$.

- $\mathrm{rand}(1, A) = 3, f(1) = 2$.
- $\mathrm{rand}(2, A) = 2, f(2) = 1$.
- $\mathrm{rand}(3, A) = 1, f(3) = 2$.

Hence the expected value of f is 1.5, which is congruent to $499\,122\,178$ modulo $998\,244\,353$.

Problem J. Roulette

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Walk Alone is playing roulette, a kind of gambling. For simplification, we assume its rules and steps as follows:

- The whole gambling process composes of many turns.
- In the i-th turn:
  - Walk Alone can choose an integer x_i and pay x_i yuan as the wager.
  - If Walk Alone wins, he will get $2x_i$ yuan from the maker, which means he gains x_i yuan in this turn. Otherwise, the maker will devour the x_i yuan he has paid, which means he loses x_i yuan in this turn. The probability that Walk Alone wins is 0.5.

Walk Alone has n yuan initially, and he wants to earn an extra m yuan. He will use the following strategy to gamble:

- In the first turn, Walk Alone pays 1 yuan as the wager, i.e., $x_1 = 1$.
- If Walk Alone wins in the $(i-1)$-th turn, then $x_i = 1$. Otherwise, $x_i = 2x_{i-1}$.
- At the beginning of each turn, if Walk Alone has at least $(n+m)$ yuan, then he gets satisfied and quits the game. Otherwise, he must pay x_i yuan as the wager, and he will have to stop gambling if he has less than x_i yuan.

As Walk Alone's good friend, Kelin wants to exhort him not to gamble. Tell him the probability that he successfully earns an extra m yuan.

Input

The only line of the input contains two integers n ($1 \le n \le 10^9$) and m ($1 \le m \le 10^9$), indicating the initial amount of money and the amount of money Walk Alone wants to earn.

Output

Output the probability that Walk Alone successfully earns an extra m yuan modulo 998 244 353. It can be shown that the answer can always be expressed as an irreducible fraction x/y, where x and y are integers and $y \not\equiv 0 \pmod{998\,244\,353}$. Output such an integer a that $0 \le a < 998\,244\,353$ and $a \cdot y \equiv x \pmod{998\,244\,353}$.

Examples

standard input	standard output
2 2	623902721
5 30	79070907

Note

In the first example, Walk Alone has 2 yuan initially.

In the first turn, he pays 1 yuan as the wager, and he has 1 yuan left. He must win the turn, or in the next turn he will have to pay 2 yuan, and he does not have enough money. The probability that he wins is 0.5.

In the second turn, he pays 1 yuan and has 2 yuan left. If he wins the turn, he will have 4 yuan, which reaches his goal. Otherwise, he will have to pay 2 yuan in the third turn, and he must win the turn to reach his goal. Hence the total probability is $0.5 \cdot (0.5 + 0.5 \cdot 0.5) = 3/8$.

Problem K. Subdivision

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

You are given a graph with n vertices and m undirected edges of length 1. You can do the following operation on the graph for arbitrary times:

- Choose an edge (u, v) and replace it by two edges, (u, w) and (w, v), where w is a newly inserted vertex. Both of the two edges are of length 1.

You need to find out the maximum number of vertices whose minimum distance to vertex 1 is no more than k.

Input

The first line contains three integers n ($1 \le n \le 10^5$), m ($0 \le m \le 2 \cdot 10^5$) and k ($0 \le k \le 10^9$).

Each of the next m lines contains two integers u and v ($1 \le u, v \le n$), indicating an edge between u and v. It is guaranteed that there are no self-loops or multiple edges.

Output

Output an integer indicating the maximum number of vertices whose minimum distance to vertex 1 is no more than k.

Examples

standard input	standard output
5 4 2 1 2 2 3 3 1 4 5	5
8 9 3 1 2 1 3 1 5 3 4 3 6 4 5 5 6 6 7 7 8	15
114 0 514	1

Problem L. Three Permutations

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

You are given three permutations a, b, c of length n. Recall that a permutation is a sequence of n integers from 1 to n, in which each number occurs exactly once.

There are three integers x, y and z. Initially, all of them equal to 1. After each second, x, y, z become a_y, b_z, c_x respectively and simultaneously (**note that the subscripts are** y, z, x).

You need to answer multiple independent queries. For each query, you are given three integers x', y' and z'. Tell Walk Alone the minimum time needed for tuple (x, y, z) to become (x', y', z') if possible.

Input

The first contains an integer n ($1 \le n \le 10^5$), indicating the length of the permutations.

Each of the next three lines contains n integers. The i-th number of the second line, the third line and the fourth line indicate a_i, b_i and c_i ($1 \le a_i, b_i, c_i \le n$), respectively.

The next line contains an integer Q ($1 \le Q \le 10^5$), indicating the number of queries.

Each of the next Q lines contains three integers x', y' and z' ($1 \le x', y', z' \le n$).

Output

For each query, output the minimum time needed for (x, y, z) to become (x', y', z') in a line, or -1 if impossible.

Examples

standard input	standard output
5 2 1 5 4 3 3 1 4 2 5 5 4 2 1 3 5 1 1 1 2 3 5 5 5 4 3 2 3 1 4 2	0 1 2 3 4
10 10 9 3 7 6 2 8 5 1 4 8 9 10 6 2 4 1 7 3 5 10 1 3 5 9 6 2 7 4 8 2 9 7 2 2 9 7	-1 89

Problem M. Water

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

Walk Alone is thirsty and he wants to drink water. He wants to drink exactly x units of water, but he does not have an appropriate measuring cup. He only has two water bottles of volume A and B respectively. He found out that he could do the following operations with the two bottles:

- Fill one of the two bottles with water.
- Spill all water from one of the two bottles.
- Drink all water from one of the two bottles.
- Transfer as much water as possible from one bottle to the other with no water overflow. Formally speaking, if the two bottles contain a and b units of water respectively, he can only transfer $\min(a, B-b)$ water from A to B or $\min(b, A-a)$ from B to A.

Walk Alone wants to do as few operations as possible. Can you tell him the minimum number of operations he should do to drink exactly x units of water?

Input

The input contains multiple test cases.

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

Each test case contains only one line with three integers A, B, x ($1 \le A, B, x \le 10^9$), the volume of the two bottles and the required volume of water he will drink, respectively.

Output

For each test case, output the minimum number of operations in one line. If it is impossible for him to drink exactly x units of water, output -1.

Example

standard input	standard output
5 2 5 1 2 3 7 4 6 3 3 13 8 12 9 6	5 6 -1 15 5

Note

In the first example, Walk Alone will do the following operations:

- Fill the second bottle.
- Transfer 2 units of water from the second bottle to the first one.
- Spill all water from the first cup.
- Transfer 2 units of water from the second bottle to the first one, after which there is exactly one unit of water in the second bottle.
- Drink all water in the second bottle.