Bubble Cup 11 - Finals [Online Mirror, Div. 2]

A. AI robots

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

In the last mission, MDCS has successfully shipped N AI robots to Mars. Before they start exploring, system initialization is required so they are arranged in a line. Every robot can be described with three numbers: position (x_i), radius of sight (r_i) and IQ (q_i).

Since they are intelligent robots, some of them will talk if they see each other. Radius of sight is inclusive, so robot can see other all robots in range $[x_i - r_i, x_i + r_i]$. But they don't walk to talk with anybody, but only with robots who have similar IQ. By similar IQ we mean that their absolute difference isn't more than K.

Help us and calculate how many pairs of robots are going to talk with each other, so we can timely update their software and avoid any potential quarrel.

Input

The first line contains two integers, numbers $N(1 \leq N \leq 10^5)$ and $K(0 \leq K \leq 20)$.

Next N lines contain three numbers each $x_i, r_i, q_i (0 \leq x_i, r_i, q_i \leq 10^9)$ — position, radius of sight and IQ of every robot respectively.

Output

Output contains only one number — solution to the problem.

Example

input

```
3 2
3 6 1
7 3 10
10 5 8
```

output

```
1
```

Note

The first robot can see the second, but not vice versa. The first robot can't even see the third. The second and the third robot can see each other and their IQs don't differ more than 2 so only one conversation will happen.

B. Hyperspace Highways

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

In an unspecified solar system, there are N planets. A space government company has recently hired space contractors to build M bidirectional Hyperspace™ highways, each connecting two different planets. The primary objective, which was to make sure that every planet can be reached from any other planet taking only Hyperspace™ highways, has been completely fulfilled. Unfortunately, lots of space contractors had friends and cousins in the Space Board of Directors of the company, so the company decided to do much more than just connecting all planets.

In order to make spending enormous amounts of space money for Hyperspace™ highways look neccessary, they decided to enforce a strict rule on the Hyperspace™ highway network: whenever there is a way to travel through some planets and return to the starting point without travelling through any planet twice, every pair of planets on the itinerary should be directly connected by a Hyperspace™ highway. In other words, the set of planets in every simple cycle induces a complete subgraph.

You are designing a Hyperspace™ navigational app, and the key technical problem you are facing is finding the minimal number of Hyperspace™ highways one needs to use to travel from planet A to planet B. As this problem is too easy for Bubble Cup, here is a harder task: your program needs to do it for Q pairs of planets.

Input

The first line contains three positive integers N $(1 \leq N \leq 100\,000)$, M $(1 \leq M \leq 500\,000)$ and Q $(1 \leq Q \leq 200\,000)$, denoting the number of planets, the number of Hyperspace™ highways, and the number of queries, respectively.

Each of the following M lines contains a highway: highway i is given by two integers u_i and v_i $(1 \leq u_i < v_i \leq N)$, meaning the planets u_i and v_i are connected by a Hyperspace™ highway. It is guaranteed that the network of planets and Hyperspace™

highways forms a simple connected graph.

Each of the following Q lines contains a query: query j is given by two integers a_j and b_j $(1 \leq a_j < b_j \leq N)$, meaning we are interested in the minimal number of Hyperspace™ highways one needs to take to travel from planet a_j to planet b_j.

Output

Output Q lines: the j-th line of output should contain the minimal number of Hyperspace™ highways one needs to take to travel from planet a_j to planet b_j.

Examples

input

```
5 7 2
1 2
1 3
1 4
2 3
2 4
3 4
1 5
1 4
2 5
```

output

```
1
2
```

input

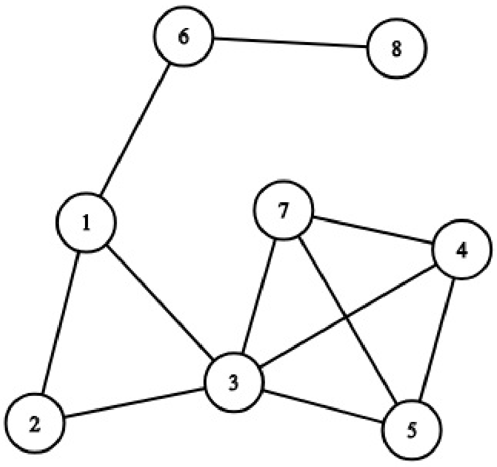
```
8 11 4
1 2
2 3
3 4
4 5
1 3
1 6
3 5
3 7
4 7
5 7
6 8
1 5
2 4
6 7
3 8
```

output

```
2
2
3
3
```

Note

The graph from the second sample:



C. Space Formula

time limit per test: 1 second

memory limit per test: 256 megabytes

input: standard input

output: standard output

Formula 1 officials decided to introduce new competition. Cars are replaced by space ships and number of points awarded can differ per race.

Given the current ranking in the competition and points distribution for the next race, your task is to calculate the best possible ranking for a given astronaut after the next race. It's guaranteed that given astronaut will have unique number of points before the race.

Input

The first line contains two integer numbers N ($1 \leq N \leq 200000$) representing number of F1 astronauts, and current position of astronaut D ($1 \leq D \leq N$) you want to calculate best ranking for (no other competitor will have the same number of points before the race).

The second line contains N integer numbers S_k ($0 \leq S_k \leq 10^8$, $k = 1...N$), separated by a single space, representing current ranking of astronauts. Points are sorted in non-increasing order.

The third line contains N integer numbers P_k ($0 \leq P_k \leq 10^8$, $k = 1...N$), separated by a single space, representing point awards for the next race. Points are sorted in non-increasing order, so winner of the race gets the maximum number of points.

Output

Output contains one integer number — the best possible ranking for astronaut after the race. If multiple astronauts have the same score after the race, they all share the best ranking.

Example

input

```
4 3
50 30 20 10
15 10 7 3
```

output

```
2
```

Note

If the third ranked astronaut wins the race, he will have 35 points. He cannot take the leading position, but he can overtake the second position if the second ranked astronaut finishes the race at the last position.

D. Interstellar battle

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

In the intergalactic empire Bubbledom there are N planets, of which some pairs are directly connected by two-way wormholes. There are $N-1$ wormholes. The wormholes are of extreme religious importance in Bubbledom, a set of planets in Bubbledom consider themselves one intergalactic kingdom if and only if any two planets in the set can reach each other by traversing the wormholes. You are given that Bubbledom is one kingdom. In other words, the network of planets and wormholes is a tree.

However, Bubbledom is facing a powerful enemy also possessing teleportation technology. The enemy attacks every night, and the government of Bubbledom retakes all the planets during the day. In a single attack, the enemy attacks every planet of Bubbledom at once, but some planets are more resilient than others. Planets are number $0, 1, \ldots, N-1$ and the planet i will fall with probability p_i. Before every night (including the very first one), the government reinforces or weakens the defenses of a single planet.

The government of Bubbledom is interested in the following question: what is the expected number of intergalactic kingdoms Bubbledom will be split into, after a single enemy attack (before they get a chance to rebuild)? In other words, you need to print the expected number of connected components after every attack.

Input

The first line contains one integer number N ($1 \le N \le 10^5$) denoting the number of planets in Bubbledom (numbered from 0 to $N-1$).

The next line contains N different real numbers in the interval $[0,1]$, specified with 2 digits after the decimal point, denoting the probabilities that the corresponding planet will fall.

The next $N-1$ lines contain all the wormholes in Bubbledom, where a wormhole is specified by the two planets it connects.

The next line contains a positive integer Q ($1 \le Q \le 10^5$), denoting the number of enemy attacks.

The next Q lines each contain a non-negative integer and a real number from interval $[0,1]$, denoting the planet the government of Bubbledom decided to reinforce or weaken, along with the new probability that the planet will fall.

Output

Output contains Q numbers, each of which represents the expected number of kingdoms that are left after each enemy attack. Your answers will be considered correct if their absolute or relative error does not exceed 10^{-4}.

Example

input

```
5
0.50 0.29 0.49 0.95 0.83
2 3
0 3
3 4
2 1
3
4 0.66
1 0.69
0 0.36
```

output

```
1.68040
1.48440
1.61740
```

E. Ancient civilizations

time limit per test: 0.5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

On the surface of a newly discovered planet, which we model by a plane, explorers found remains of two different civilizations in various locations. They would like to learn more about those civilizations and to explore the area they need to build roads between some of locations. But as always, there are some restrictions:

1. Every two locations of the same civilization are connected by a unique path of roads
2. No two locations from different civilizations may have road between them (explorers don't want to accidentally mix civilizations they are currently exploring)
3. Roads must be **straight line segments**
4. Since intersections are expensive to build, they don't want any two roads to intersect (that is, only common point for any two roads may be at some of locations)

Obviously all locations are different points in the plane, but explorers found out one more interesting information that may help you – no three locations lie on the same line!

Help explorers and find a solution for their problem, or report it is impossible.

Input

In the first line, integer n $(1 \leq n \leq 10^3)$ - the number of locations discovered.

In next n lines, three integers x, y, c $(0 \leq x, y \leq 10^4, c \in \{0, 1\})$ - coordinates of the location and number of civilization it belongs to.

Output

In first line print number of roads that should be built.

In the following lines print all pairs of locations (their 0-based indices) that should be connected with a road.

If it is not possible to build roads such that all restrictions are met, print "`Impossible`". You should not print the quotation marks.

Example

input

```
5
0 0 1
1 0 0
0 1 0
1 1 1
3 2 0
```

output

```
3
1 4
4 2
3 0
```

F. Splitting money

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

After finding and moving to the new planet that supports human life, discussions started on which currency should be used. After long negotiations, Bitcoin was ultimately chosen as the universal currency.

These were the great news for Alice, whose grandfather got into Bitcoin mining in 2013, and accumulated a lot of them throughout the years. Unfortunately, when paying something in bitcoin everyone can see how many bitcoins you have in your public address wallet.

This worried Alice, so she decided to split her bitcoins among multiple different addresses, so that every address has at most x satoshi (1 bitcoin = 10^8 satoshi). She can create new public address wallets for free and is willing to pay f fee in satoshi per transaction to ensure acceptable speed of transfer. The fee is deducted from the address the transaction was sent from. Tell Alice how much total fee in satoshi she will need to pay to achieve her goal.

Input

First line contains number N ($1 \leq N \leq 200\,000$) representing total number of public addresses Alice has.

Next line contains N integer numbers a_i ($1 \leq a_i \leq 10^9$) separated by a single space, representing how many satoshi Alice has in her public addresses.

Last line contains two numbers x, f ($1 \leq f < x \leq 10^9$) representing maximum number of satoshies Alice can have in one address, as well as fee in satoshies she is willing to pay per transaction.

Output

Output one integer number representing total fee in satoshi Alice will need to pay to achieve her goal.

Example

input

```
3
13 7 6
6 2
```

output

```
4
```

Note

Alice can make two transactions in a following way:

0. 13 7 6 (initial state)

1. 6 7 6 5 (create new address and transfer from first public address 5 satoshies)

2. 6 4 6 5 1 (create new address and transfer from second address 1 satoshi)

Since cost per transaction is 2 satoshies, total fee is 4.

G. Space Isaac

time limit per test: 1.5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Everybody seems to think that the Martians are green, but it turns out they are metallic pink and fat. Ajs has two bags of distinct nonnegative integers. The bags are disjoint, and the union of the sets of numbers in the bags is $\{0,1,\ldots,M-1\}$, for some positive integer M. Ajs draws a number from the first bag and a number from the second bag, and then sums them modulo M.

What are the residues modulo M that Ajs cannot obtain with this action?

Input

The first line contains two positive integer N ($1 \leq N \leq 200\,000$) and M ($N+1 \leq M \leq 10^9$), denoting the number of the elements in the first bag and the modulus, respectively.

The second line contains N nonnegative integers $a_1, a_2, \ldots, a_N$ ($0 \leq a_1 < a_2 < \ldots < a_N < M$), the contents of the first bag.

Output

In the first line, output the cardinality K of the set of residues modulo M which Ajs cannot obtain.

In the second line of the output, print K space-separated integers greater or equal than zero and less than M, which represent the residues Ajs cannot obtain. The outputs should be sorted in increasing order of magnitude. If K=0, do not output the second line.

Examples

input

```
2 5
3 4
```

output

```
1
2
```

input

```
4 1000000000
5 25 125 625
```

output

```
0
```

input

```
2 4
1 3
```

output

```
2
0 2
```

Note

In the first sample, the first bag and the second bag contain $\{3,4\}$ and $\{0,1,2\}$, respectively. Ajs can obtain every residue modulo 5 except the residue 2: $4+1 \equiv 0$, $4+2 \equiv 1$, $3+0 \equiv 3$, $3+1 \equiv 4$ modulo 5. One can check that there is no choice of elements from the first and the second bag which sum to 2 modulo 5.

In the second sample, the contents of the first bag are $\{5,25,125,625\}$, while the second bag contains all other nonnegative integers with at most 9 decimal digits. Every residue modulo $1\,000\,000\,000$ can be obtained as a sum of an element in the first bag and an element in the second bag.

H. Palindrome Pairs

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

After learning a lot about space exploration, a little girl named Ana wants to change the subject.

Ana is a girl who loves palindromes (string that can be read the same backwards as forward). She has learned how to check for a given string whether it's a palindrome or not, but soon she grew tired of this problem, so she came up with a more interesting one and she needs your help to solve it:

You are given an array of strings which consist of only small letters of the alphabet. Your task is to find **how many** palindrome pairs are there in the array. A palindrome pair is a pair of strings such that the following condition holds: **at least one** permutation of the concatenation of the two strings is a palindrome. In other words, if you have two strings, let's say "aab" and "abcac", and you concatenate them into "aababcac", we have to check if there exists a permutation of this new string such that it is a palindrome (in this case there exists the permutation "aabccbaa").

Two pairs are considered different if the strings are located on **different indices**. The pair of strings with indices (i, j) is considered **the same** as the pair (j, i).

Input

The first line contains a positive integer N ($1 \le N \le 100\,000$), representing the length of the input array.

Eacg of the next N lines contains a string (consisting of lowercase English letters from 'a' to 'z') — an element of the input array.

The total number of characters in the input array will be less than $1\,000\,000$.

Output

Output one number, representing **how many palindrome pairs** there are in the array.

Examples

input

```
3
aa
bb
cd
```

output

```
1
```

input

```
6
aab
abcac
dffe
ed
aa
aade
```

output

```
6
```

Note

The first example:

1. aa $+$ bb $\to$ abba.

The second example:

1. aab $+$ abcac $=$ aababcac $\to$ aabccbaa
2. aab $+$ aa $=$ aabaa
3. abcac $+$ aa $=$ abcacaa $\to$ aacbcaa
4. dffe $+$ ed $=$ dffeed $\to$ fdeedf
5. dffe $+$ aade $=$ dffeaade $\to$ adfaafde
6. ed $+$ aade $=$ edaade $\to$ aeddea

I. Say Hello

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Two friends are travelling through Bubble galaxy. They say "Hello!" via signals to each other if their distance is smaller or equal than d_1 and

- it's the first time they speak to each other or
- at some point in time after their last talk their distance was greater than d_2.

We need to calculate how many times friends said "Hello!" to each other. For N moments, you'll have an array of points for each friend representing their positions at that moment. A person can stay in the same position between two moments in time, but if a person made a move we assume this movement as movement with constant speed in constant direction.

Input

The first line contains one integer number N ($2 \le N \le 100\,000$) representing number of moments in which we captured positions for two friends.

The second line contains two integer numbers d_1 and d_2 $(0 < d_1 < d_2 < 1000)$.

The next N lines contains four integer numbers A_x, A_y, B_x, B_y ($0 \le A_x, A_y, B_x, B_y \le 1000$) representing coordinates of friends A and B in each captured moment.

Output

Output contains one integer number that represents how many times friends will say "Hello!" to each other.

Example

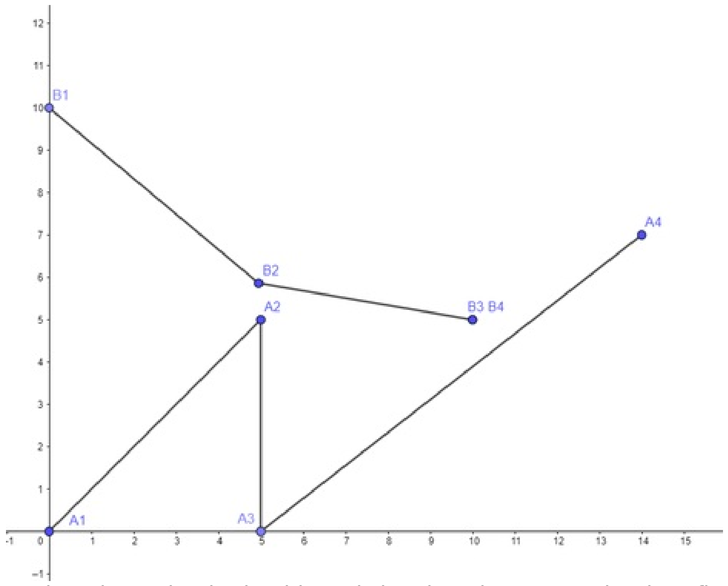
input

```
4
2 5
0 0 0 10
5 5 5 6
5 0 10 5
14 7 10 5
```

output

```
2
```

Note



Explanation: Friends should send signals 2 times to each other, first time around point $A2$ and $B2$ and second time during A's travel from point $A3$ to $A4$ while B stays in point $B3 = B4$.

J. Self-exploration

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Being bored of exploring the Moon over and over again Wall-B decided to explore something he is made of — binary numbers. He took a binary number and decided to count how many times different substrings of length two appeared. He stored those values in c_{00}, c_{01}, c_{10} and c_{11}, representing how many times substrings `00`, `01`, `10` and `11` appear in the number respectively. For example:

$10111100 \to c_{00} = 1,\ c_{01} = 1,\ c_{10} = 2,\ c_{11} = 3$

$10000 \to c_{00} = 3,\ c_{01} = 0,\ c_{10} = 1,\ c_{11} = 0$

$10101001 \to c_{00} = 1,\ c_{01} = 3,\ c_{10} = 3,\ c_{11} = 0$

$1 \to c_{00} = 0,\ c_{01} = 0,\ c_{10} = 0,\ c_{11} = 0$

Wall-B noticed that there can be multiple binary numbers satisfying the same c_{00}, c_{01}, c_{10} and c_{11} constraints. Because of that he wanted to count how many binary numbers satisfy the constraints c_{xy} given the interval $[A, B]$. Unfortunately, his processing power wasn't strong enough to handle large intervals he was curious about. Can you help him? Since this number can be large print it modulo $10^9 + 7$.

Input

First two lines contain two positive binary numbers A and B ($1 \leq A \leq B < 2^{100\,000}$), representing the start and the end of the interval respectively. Binary numbers A and B have no leading zeroes.

Next four lines contain decimal numbers c_{00}, c_{01}, c_{10} and c_{11} ($0 \leq c_{00}, c_{01}, c_{10}, c_{11} \leq 100\,000$) representing the count of two-digit substrings `00`, `01`, `10` and `11` respectively.

Output

Output one integer number representing how many binary numbers in the interval $[A, B]$ satisfy the constraints mod $10^9 + 7$.

Examples

input

```
10
1001
0
0
1
1
```

output

```
1
```

input

```
10
10001
1
2
3
4
```

output

```
0
```

Note

Example 1: The binary numbers in the interval $[10, 1001]$ are $10, 11, 100, 101, 110, 111, 1000, 1001$. Only number 110 satisfies the constraints: $c_{00} = 0, c_{01} = 0, c_{10} = 1, c_{11} = 1$.

Example 2: No number in the interval satisfies the constraints