Count the Floors! (elevator)

Edoardo heard some strange noises coming from his elevator, so he has been tracking which trips the elevator has made in order to find where the problem is. At the beginning, the elevator was at floor F_0. Then, the elevator made $N-1$ trips, reaching after the i-th trip the floor F_i $(1 \le i \le N-1)$. Edoardo wants to know which is the most visited floor, that is the floor where the elevator passed the most number of times during all the trips. Please note that the floors where the elevator began or ended its trips are counted only once and not twice.

Figure 1: The elevator in Edoardo's house.

Edoardo, however, is not really sure that he tracked the elevator correctly, so he has made Q consecutive changes to the trips. In particular, the j-th time he change the value of F_{P_j} to V_j. Can you tell Edoardo which is the most visited floor after each change and how many times it has been visited?

> ☞ Among the attachments of this task you may find a template file `elevator.*` with a sample incomplete implementation.

Input

The first line contains the only integer N. The second line contains N integers F_i. The third line contains the only integer Q. Then Q lines follow, the j-th of which contains two integers P_j and V_j.

Output

For each one of the Q changes, you need to write a single line with two integers: the most visited floor and how many times it has been visited. If there are multiple floors which have been visited the same number of times, write the smallest one.

Constraints

- $2 \le N \le 100\,000$.

- $1 \leq F_i \leq 200\,000$ for each $i = 0 \ldots N-1$.
- $1 \leq Q \leq 100\,000$.
- $0 \leq P_j \leq N-1$ for each $j = 0 \ldots Q-1$.
- $1 \leq V_j \leq 200\,000$ for each $j = 0 \ldots Q-1$.
- F_i are distinct at the beginning and after every update.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

– **Subtask 1** (0 points)	Examples.
– **Subtask 2** (25 points)	$1 \leq N, Q, F_i, V_j \leq 200$.
– **Subtask 3** (35 points)	$1 \leq N, Q \leq 1\,000$, $1 \leq F_i, V_j \leq 10\,000$.
– **Subtask 4** (40 points)	No additional limitations.

Examples

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

Explanation

In the **first sample case**, after all the queries, $F = [3, 1, 5, 4, 2]$. This means that:

- the first trip visits floors 3, 2 and 1;
- the second trip visits floors 2, 3, 4 and 5 (floor 1 was already counted in the previous trip);
- the third trip visits floor 4;
- the last trip visits floors 3 and 2;

This means that the most visited floor is 2, which is visited three times. Note that also floor 3 has been visited three times, but it is not the smallest.

Do Not Gather! (gatherings)

As vaccinations against COVID-19 are starting all over the world, people seem to keep forgetting the one and only piece of advice they have been given in the last months: "Do Not Gather!".

To receive their dose, people usually queue up outside the administration center and form a line. Often, though, the minimum interpersonal distance D (measured in centimeters) imposed by law is not respected, enabling possible infections.

Figure 1: An example of how to *not* queue up properly.

Luca was walking and accidentally noted the troubling situation: out of N people in queue at positions P_i, also measured in centimeters starting from the vaccination center, many seem to be in a risky position. How many pairs of people do not respect the distance, i.e. the distance between them is less than the required distance?

☞ Among the attachments of this task you may find a template file `gatherings.*` with a sample incomplete implementation.

Input

The first line contains two integers, N and D. The second line contains N integers P_i.

Output

You need to write a single line with an integer: the number of pairs of people who do not respect the minimum distance.

✍ The answer may not fit into a 32-bit integer: use `long long` in C/C++ and `int64` in Pascal in order to avoid integer overflow. The provided templates are already properly set.

Constraints

- $1 \leq N \leq 100\,000$.
- $1 \leq D \leq 10^9$.
- $0 \leq P_i \leq 10^9$ for each $i = 0 \ldots N-1$.
- Positions are all distinct and are listed from the nearest to the farthest (i.e., $P_i < P_{i+1}$ for each $i = 0 \ldots N-2$).

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

– **Subtask 1** (0 points) Examples.

– **Subtask 2** (30 points) $D \leq 10\,000, N \leq 10\,000$.

– **Subtask 3** (30 points) $D \leq 10\,000$.

– **Subtask 4** (40 points) No additional limitations.

Examples

input	output
`4 100` `20 120 200 300`	`1`
`4 200` `0 100 150 200`	`5`

Explanation

In the **first sample case** the second and the third person are at distance 200-120 = 80 centimeters which is below 100. All other pairs respect the distance.

In the **second sample case** there are five problematic pairs: the first and the second person (distance 100), the first and the third one (distance 150), the second and the third one (distance 50), the second and the fourth one (distance 100) and the third and the fourth one (distance 50).

Back to the Himalayas (himalaya)

Marco is a great fan of the Himalaya mountains. For his next trip there, Marco wants to go through a sequence of N peaks on a straight line of varying height H_i (for $i = 0 \dots N-1$), moving from the left to the right only using his beloved bobsleigh!

Figure 1: The Himalaya mountains.

More precisely, when starting with his bobsleigh in a peak, Marco runs up to a speed of $V m/s$, then lets its bobsleigh proceed on free fall, as long as it has enough energy to keep going. More precisely, Marco starts with a *kinetic energy* of $\frac{1}{2}MV^2$, where M is the mass of Marco and his bobsleigh. When moving from the i-th peak to the $i+1$-th peak, if $H_i > H_{i+1}$ then Marco will gain $MG(H_i - H_{i+1})$ kinetic energy, otherwise he will lose $MG(H_{i+1} - H_i)$ kinetic energy, where $G = 10m/s^2$ is the gravitational acceleration. The bobsleigh can keep moving as long as the kinetic energy does not become negative.

In order to plan his trip, Marco needs to know how far to the right will be able to go with a single free fall, starting from *each* of the N peaks.

> ☞ Among the attachments of this task you may find a template file `himalaya.*` with a sample incomplete implementation.

Input

The first line contains the three integers N, M, V. The second line contains N integers H_i.

Output

You need to write a single line with N integers: the indices (from 0 to $N-1$) of the farthest peaks to the right that can be reached starting from each of the N peaks.

Constraints

- $1 \leq N \leq 500\,000$.
- $1 \leq M \leq 10^9$.
- $1 \leq V \leq 40\,000$.
- $1 \leq H_i \leq 10^9$ for each $i = 0 \dots N-1$.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

– **Subtask 1** (0 points) Examples.

– **Subtask 2** (30 points) $N \leq 1000$, $V \leq 400$.

– **Subtask 3** (30 points) $N \leq 1000$.

– **Subtask 4** (40 points) No additional limitations.

Examples

input	output
`1 50 100` `1000`	`0`
`9 2 7` `3 2 1 5 3 3 8 2 1`	`5 2 2 5 5 5 8 8 8`

Explanation

In the **first sample case**, there is only one peak, which is the farthest to the right that can be reached starting from that peak itself.

In the **second sample case**, let's see why the farthest reachable peak starting from 0 is peak 5. Marco starts with a kinetic energy of $\frac{1}{2}2 \cdot 7^2 = 49$.

- Moving from peak 0 to 1, his energy increases by $MG(H_0 - H_1) = 2 \cdot 10 \cdot 1 = 20$, becoming 69.
- Moving from peak 1 to 2, his energy increases by $MG(H_1 - H_2) = 2 \cdot 10 \cdot 1 = 20$, becoming 89.
- Moving from peak 2 to 3, his energy decreases by $MG(H_3 - H_2) = 2 \cdot 10 \cdot 4 = 80$, becoming 9.
- Moving from peak 3 to 4, his energy increases by $MG(H_3 - H_4) = 2 \cdot 10 \cdot 2 = 40$, becoming 49.
- Moving from peak 4 to 5, his energy stays the same.
- Moving from peak 5 to 6, his energy would decrease by $MG(H_6 - H_5) = 2 \cdot 10 \cdot 5 = 100$, becoming $49 - 100 = -51$ which is below zero. Thus, Marco cannot reach peak 6 from peak 0.

Perfect Hyperrectangle (hyperrectangle)

After decades of research, Giorgio is finally about to solve the mystery of the *perfect N-hyperrectangle*! This legendary geometrical structure is defined by its lower coordinate L_i and higher coordinate H_i for every axis $i = 0 \dots N-1$. In an ancient book, Giorgio has found the coordinates he was looking for.

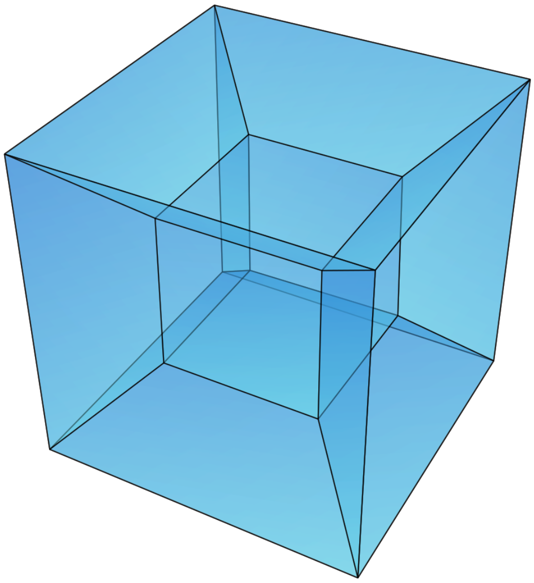

Figure 1: Artistic representation of a 4-hyperrectangle.

Unfortunately, the book has been shredded over the centuries, mixing up the $2N$ numbers into a single list V_i for $i = 0 \dots 2N-1$, and Giorgio needs to reconstruct the correct arrangement of the $2N$ numbers into coordinates L_i and H_i. This arrangement has to produce the *largest hypervolume*, computed as:

$$(H_0 - L_0) \times \dots (H_{N-1} - L_{N-1})$$

Furthermore, among arrangements with the largest hypervolume, the sequence $(L_0, \dots, L_{N-1}, H_0, \dots, H_{N-1})$ has to be *lexicographically minimum*.

✍ A sequence $(a_1, \dots, a_l)$ is *smaller* than another $(b_1, \dots, b_l)$ if there is a k such that $a_i = b_i$ for all $i = 1 \dots k-1$ and $a_k < b_k$.

☞ Among the attachments of this task you may find a template file `hyperrectangle.*` with a sample incomplete implementation.

Input

The first line contains the only integer N. The second line contains $2N$ integers V_i.

Output

You need to write a two lines with N integers each: the lower coordinates L_i and higher coordinates H_i, respectively.

Constraints

- $1 \leq N \leq 100\,000$.
- $0 \leq V_i \leq 10^9$ for each $i = 0 \ldots N-1$.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

– **Subtask 1** (0 points) Examples.

– **Subtask 2** (30 points) $N \leq 5$, $V_i \leq 50$.

– **Subtask 3** (50 points) $N \leq 1000$.

– **Subtask 4** (20 points) No additional limitations.

Examples

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

Explanation

In the **first sample case**, there is only one possible arrangement.

In the **second sample case**, the ordering is lexicographically minimal among those with maximal hypervolume $(7-4) \times (9-5) = 3 \times 4 = 12$.

Some Infection (`infection`)

In another planet, totally different and not at all related to Earth, there is a sneaky epidemic that is infecting most of the population. The symptoms are not always evident, therefore it's spreading pretty much unnoticed.

People are arranged in a grid of N rows and M columns. This virus is very infectious, if a person is infected at time t, then at time $t+1$ all 4 neighbours will be infected as well. Furthermore, the older the person infected, the worse are the effects of the illness! You know the age A_{ij} of each person.

Figure 1: Avoid this please!

The infection starts at time 0 from a person in the grid, and it evolves until it is noticed. The virus gets noticed when a person on the border is infected; at that point, Mr *Withyou*, the President of the World, declares a *global lockdown*, stopping the spreading of the virus.

You don't know who was the *patient zero*, the one that started the epidemic, but since you are a strong believer in *Murphy's Law*[1], you simply assume it was the one that **maximizes the sum of the ages** of the people infected when the virus is noticed.

Mr Withyou is quite busy fending off all his fangirls: help him find the patient zero!

> ☞ Among the attachments of this task you may find a template file `infection.*` with a sample incomplete implementation.

Input

The first line contains two integers N and M. The following N lines contain M integers each, the age A_{ij} of each person.

Output

You need to write a single line with two integers r ($0 \le r < N$) and c ($0 \le c < M$): the row and the column of the *patient zero*.

[1] *"Anything that can go wrong, will go wrong"*

Constraints

- $1 \leq N, M \leq 1000$.
- $0 \leq A_{ij} \leq 10^9$ for each $i = 0 \ldots N-1$ and $j = 0 \ldots M-1$.
- If more than a person could be the *patient zero*, you can print any of them.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

– **Subtask 1** (0 points) Examples.

– **Subtask 2** (7 points) $N \leq 3$.

– **Subtask 3** (9 points) All the person have the same age.

– **Subtask 4** (21 points) $N, M \leq 50$.

– **Subtask 5** (28 points) $N, M \leq 500$.

– **Subtask 6** (35 points) No additional limitations.

Examples

input	output
4 7 1 8 7 2 6 1 1 2 4 8 5 1 2 9 9 8 5 1 0 8 9 9 9 8 2 0 1 1	2 1
2 5 1 2 6 8 4 3 5 1 4 8	0 3

Explanation

In the **first sample case** the person that maximizes the sum of ages is in the third row, second column.

$$\begin{array}{ccccccc} 1 & 8 & 7 & 2 & 6 & 1 & 1 \\ 2 & 4 & 8 & 5 & 1 & 2 & 9 \\ 9 & 8 & 5 & 1 & 0 & 8 & 9 \\ 9 & 9 & 8 & 2 & 0 & 1 & 1 \end{array}$$

In the **second sample case** since every person is on the border, as soon as the infection starts it is immediately stopped. The solution is therefore one of the two 8-year-old persons.

Improve the Team! (lootboxes)

William was very bored during lockdown, so he decided to download a soccer game on his smartphone. Since then, he has played a lot of matches and he has earned X coins in total. However, winning is becoming increasingly difficult so he needs to improve his team with new players.

Figure 1: One of the possible loot boxes.

In order to do so, he wants to spend the coins he has earned to open some of the N loot boxes available. The i-th loot box has probability P_i of containing a good player and costs Q_i coins. Each probability P_i is an integer number between zero and one hundred, representing the probability as a percentage. Can you help William decide which loot boxes to open in order to maximize the expected number of good players he can get?

☞ Among the attachments of this task you may find a template file `lootboxes.*` with a sample incomplete implementation.

Input

The first line contains two integers N and X. Then N lines follow, the i-th line contains two integers P_i and Q_i.

Output

You need to write a single line with an integer: the maximum expected number of good players William might find in the loot boxes, expressed as a percentage.

Constraints

- $1 \le N \le 5000$.
- $1 \le X \le 10\,000$.
- $0 \le P_i \le 100$ for each $i = 0 \ldots N-1$.

- $1 \leq Q_i \leq 10\,000$ for each $i = 0 \ldots N-1$.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

– **Subtask 1** (0 points) Examples.

– **Subtask 2** (20 points) $N \leq 20$.

– **Subtask 3** (45 points) $N \cdot X \leq 10^6$.

– **Subtask 4** (35 points) No additional limitations.

Examples

input	output
6 3 70 1 54 3 20 1 9 1 20 2 33 1	123

Explanation

In the **first sample case**, the maximum expected number of good players William can get is 1.23, or 123% as percentage. In order to do so, he can open the first, third and sixth loot boxes paying a total of $1 + 1 + 1 = 3$ coins. In this case, the expected value is $1 \cdot 70\% + 1 \cdot 20\% + 1 \cdot 33\% = 123\%$.

Prime Pancakes (pancakes)

Stefan just bought a pancake shop, and he's going to sell N different kinds of pancake there. Each pancake is labeled with an integer number L_i. The label is a positive integer up to 6 digits long ($1 \leq L_i \leq 999\,999$). Different kinds of pancake can sometimes have the same label.

In a few hours the store will be open for business, but Stefan just remembered that he still hasn't decided the prices of each pancake type! In a hurry, Stefan decided to use his imagination and came up with a really interesting algorithm for deciding the price of a pancake.

Since Stefan loves prime numbers, the price is going to depend on "how prime" a pancake is:

- The starting price of a pancake is B euro.
- If the label's value L_i is prime, the price is increased by L_i + a "primeness bonus" of P euro.
- For each digit d of the label: if d is prime (that is, if it's either 2, 3, 5, or 7) then the price of the pancake will be increased by a corresponding fixed amount (D_2, D_3, D_5, or D_7 euro).
- If the sum of all digits $\sum d$ of the label is prime, the pancake's price is increased by $\sum d$ euro.
- If the product of all digits $\prod d$ of the label is prime, the pancake's price is increased by $\prod d$ euro.

Even though this algorithm is quite convoluted, sadly it doesn't always yield prices that are high enough to make a profit. Noticing this, Stefan decided that it's okay to change the pancakes' labels slightly, to increase their price.

You will be allowed to change **at most one digit** from each label, but on one condition: the number of digits should stay the same, so you can't change the most significant digit to a zero, because that would reduce the total number of digits.

Help Stefan compute the **sum of all pancakes' prices** after changing up to one digit from each label!

☞ Among the attachments of this task you may find a template file `pancakes.*` with a sample incomplete implementation.

Input

The first line contains the only integer N. The second line contains N integers L_i. The third line contains two integers B and P. The fourth line contains the four values D_2, D_3, D_5, and D_7.

Output

You need to write a single line with an integer: the sum of all pancakes' prices after changing up to one digit from each label.

Constraints

- $1 \leq N \leq 100\,000$.
- $1 \leq L_i \leq 999\,999$ for each $i = 0 \ldots N-1$.
- $1 \leq B, P \leq 1\,000\,000$.
- $1 \leq D_2, D_3, D_5, D_7 \leq 1\,000\,000$.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

– **Subtask 1** (0 points) Examples.

– **Subtask 2** (30 points) $N \leq 1000$.

– **Subtask 3** (70 points) No additional limitations.

Examples

input	output
4 12 89 941 101 5 7 8 5 3 9	1907

Explanation

In the **first sample case**, we can do these changes:

- First label: $12 \to$ **17**, for a price of: 45.
- Second label: stays **89**, for a price of: 118.
- Third label: $941 \to$ **991**, for a price of: 1022.
- Fourth label: $101 \to$ **701**, for a price of: 722.

The sum of all prices, after the changes are done, is 1907.

Guess The Sign (`sign`)

Edoardo and Giorgio are playing the “Guess the Sign” game. It works like this: the first player chooses two integers A and B, and the second one has to guess if the product of all integers in the $[A, B]$ range is positive, negative, or null.

For example, imagine that Edoardo chooses $A = 1$ and $B = 5$, Giorgio then needs to quickly guess that the product of all integers in the $[1, 5]$ range **is positive**, because $1 \times 2 \times 3 \times 4 \times 5 = 120$, which indeed is positive. Then, during his turn, Giorgio might decide to choose $A = -1$ and $B = 1$, and Edoardo would need to quickly guess that the product of all integers in the $[-1, 1]$ range is null, because $-1 \times 0 \times 1$ is equal to zero. (The game usually ends whenever a player makes the first mistake.)

The game is very fast-paced, therefore guesses should be made very quickly. Today Giorgio and Edoardo decided to play exactly T turns and, in order to quickly verify their answers, they asked you to write a program.

> ☞ Among the attachments of this task you may find a template file `sign.*` with a sample incomplete implementation.

Input

The first line contains the integer T, the number of turns. Each of the next T lines describes a turn, and contains two integers A and B separated by a space.

Output

You need to write T lines, one for every turn, each containing exactly one character: ‘`+`’, ‘`-`’ or ‘`0`’ (all without quotes) depending on the sign of the product of the integers in range chosen during the corresponding turn.

Constraints

- $1 \leq T \leq 100$.
- $-10^{18} \leq A \leq B \leq 10^{18}$.

Scoring

Your program will be tested against several test cases, and your score will proportional to the number of correctly solved test cases.

Note: the sample test cases are not part of the official test cases!

Examples

input	output
2 1 5 -1 1	+ 0
1 -10 -10	-

Explanation

The **first sample case** contains the two turns described in the problem statement.

The **second sample case** has only one turn, and in that turn the player chooses a range formed by one integer only: -10. The product is simply -10.

Johnnie Walker (`walker`)

Johnnie Walker is now in Bucharest! Even though it's getting late, he still wants to go for a walk. The neighborhood Johnnie stays in has N intersections of its streets, labeled from 1 to N. Johnnie's house is really close to intersection 1: the walk has to **start** and **end** there.

Figure 1: Beware when going for a stroll in some cities: you might get lost!

Given his prodigious walking abilities, at any time Johnnie can go from any intersection to any other one in approximately one minute (he is capable of adjusting the walking speed appropriately!). The only thing our hero hates is standing still and getting cold: that's why he will never pause at intersections.

After walking along different paths for a couple of evenings for K minutes, Johnnie wonders how many such paths of exactly K minutes are possible. Compute the answer for him!

> ☞ Among the attachments of this task you may find a template file `walker.*` with a sample incomplete implementation.

Input

The first and only line contains two integers: N and K.

Output

You need to write a single line with an integer: the number of different paths that Johnnie can take. Since this number may be large, report it modulo 666 013.

> ✍ The *modulo* operation ($a \bmod m$) can be written in C/C++/Python as `(a % m)` and in Pascal as `(a mod m)`. To avoid the *integer overflow* error, remember to reduce all partial results through the modulus, and not just the final result!

Constraints

- $1 \leq N \leq 10^9$.
- $1 \leq K \leq 10^{18}$.
- Johnnie may pass more than once in a certain intersection during his walk.
- Two paths are considered different if there is at least one position in which the corresponding intersections differ.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

– **Subtask 1** (0 points) Examples.

– **Subtask 2** (30 points) $N \leq 5$, $K \leq 10$.

– **Subtask 3** (30 points) $N \cdot K \leq 2\,000,000$.

– **Subtask 4** (20 points) $K \leq 1\,000,000$.

– **Subtask 5** (20 points) No additional limitations.

Examples

input	output
4 3	6
5 3	12

Explanation

In the **first sample case** the six possible paths are:

- $1 \to 2 \to 3 \to 1$
- $1 \to 2 \to 4 \to 1$
- $1 \to 3 \to 2 \to 1$
- $1 \to 3 \to 4 \to 1$
- $1 \to 4 \to 2 \to 1$
- $1 \to 4 \to 3 \to 1$

In the **second sample case** there are 12 different possible paths that can be completed in 3 minutes starting and ending at intersection 1.