Problem A. Ski race

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

During the skiing competion n skiers start the race one after another using 1 minute interval. Each sportsman moves with a constant speed and it takes exactly w_i minutes for the i-th skier to travel 1 kilometer.

The total distance of the race is L kilometers for all skiers. We say that the i-th skier has *overtaken* the j-th one, if the i-th started the race after the j-th, but finished strictly before him.

Given the information about the race, count the total number of overtakings in it.

Input

The first line of the input contains two integers n and L ($1 \le n \le 500\,000$, $1 \le L \le 10^9$) — the number of skiers participating and the length of the race.

There are n integers w_i in the second line of the input. The i-th of them is equal to the number of minutes i-th skier needs to travel 1 kilometer. First number corresponds to the first skier to start, second one corresponds to the second skier to start and so on.

Output

Print one integer — the total number of overtakings that will take place during the race.

Examples

standard input	standard output
2 1 20 19	0
5 3 3 6 2 4 1	7

Problem B. Inverse RMQ

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

Suppose we have an array $a_1, a_2, \ldots, a_n$. Define $RMQ(i, j)$ as the minimum value a_k for $k \in [i, j]$. Given some pairs of queries and answers, you are to restore an array a_i.

Input

The first line of the input contains two integers n and m ($1 \le n, m \le 100\,000$) — the length of an array and the number of range minimum queries respectively.

Then follow m lines describing the queries. Each line contains three integers i, j and q ($1 \le i \le j \le n$, $-2^{31} \le q \le 2^{31} - 1$), meaning that $RMQ(i, j) = q$.

Output

If there is no such array, print «`inconsistent`» in the only line of the output.

Otherwise, first line of the output must contain the word «`consistent`». The second line must contain the corresponding array a_i ($-2^{31} \le a_i \le 2^{31} - 1$) of length n. If there are multiple correct solution, print any.

Examples

standard input	standard output
3 2 1 2 1 2 3 2	consistent 1 2 2
3 3 1 2 1 1 1 2 2 3 2	inconsistent

Problem C. Smart Cheater

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

I guess there's not much point in reminding you that Nvodsk winters aren't exactly hot. That increased the popularity of the public transport dramatically. The route of bus 62 has exactly n stops (stop 1 goes first on its way and stop n goes last). The stops are positioned on a straight line and their coordinates are $0 = x_1 < x_2 < \cdots < x_n$.

Each day exactly m people use bus 62. For each person we know the number of the stop where he gets on the bus and the number of the stop where he gets off the bus. A ticket from stop a to stop b ($a < b$) costs $x_b - x_a$ rubles. However, the conductor can choose no more than one segment NOT TO SELL a ticket for. We mean that conductor should choose C and D (C <= D) and sell a ticket for the segments $[A, C]$ and $[D, B]$, or not sell the ticket at all. The conductor and the passenger divide the saved money between themselves equally. The conductor's "untaxed income" is sometimes interrupted by inspections that take place as the bus drives on some segment of the route located between two consecutive stops. The inspector fines the conductor by c rubles for each passenger who doesn't have the ticket for this route's segment.

You know the coordinated of all stops x_i; the numbers of stops where the i-th passenger gets on and off, a_i and b_i ($a_i < b_i$); the fine c; and also p_i — the probability of inspection on segment between the i-th and the $i+1$-th stop. The conductor asked you to help him make a plan of selling tickets that maximizes the mathematical expectation of his profit.

Input

The first line contains three integers n, m and c ($2 \le n \le 150\,000$, $1 \le m \le 300\,000$, $1 \le c \le 10\,000$).

The next line contains n integers x_i ($0 \le x_i \le 10^9$, $x_1 = 0$, $x_i < x_{i+1}$) — the coordinates of the stops on the bus's route.

The third line contains $n - 1$ integer p_i ($0 \le p_i \le 100$) — the probability of inspection in percents on the segment between stop i and stop $i + 1$.

Then follow m lines that describe the bus's passengers. Each line contains exactly two integers a_i and b_i ($1 \le a_i < b_i \le n$) — the numbers of stops where the i-th passenger gets on and off.

Output

Print the single real number — the maximum expectation of the conductor's profit. Your answer will be considered correct if its absolute or relative error does not exceed 10^{-6}.

Namely: let's assume that your answer is a, and the answer of the jury is b. The checker program will consider your answer correct, if $\frac{|a-b|}{\max(1,b)} \le 10^{-6}$.

Examples

standard input	stdout
3 3 10 0 10 100 100 0 1 2 2 3 1 3	90.000000000
10 8 187 0 10 30 70 150 310 630 1270 2550 51100 13 87 65 0 100 44 67 3 4 1 10 2 9 3 8 1 5 6 10 2 7 4 10 4 5	76859.990000000

Note

A comment to the first sample:

The first and third passengers get tickets from stop 1 to stop 2. The second passenger doesn't get a ticket. There always is inspection on the segment 1-2 but both passengers have the ticket for it. There never is an inspection on the segment 2-3, that's why the second passenger gets away with the cheating. Our total profit is $(0 + 90/2 + 90/2) = 90$.

Problem D. Gym Class

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	512 mebibytes

Feoctist Vsevolodovich is an old school teacher of physical culture. He thinks that the best way for his students to stand in a row is to stand in order of increasing height. To achieve this he first asks them to take arbitrary places in a row, and then swaps some pairs of **neighbors**, until the goal is achieved.

There are n students attending the current lesson. After the initial build-up the height of i-th student in a row is equal to h_i. You may assume that all h_i are distinct integers in range from 1 to n. Feoctist Vsevolodovich thinks that the row is sorted only if the first place is occupied by the student of height equal to 1, the second place is occupied by the student of height 2 and so on.

The teacher never does meaningless operations, i.e. he never swaps two students at position i and $i + 1$ if $h_i < h_{i+1}$. From the other hand, he likes to sort students, so he always chooses the longest possible sequence of operations.

Sasha likes to play volleyball and he understands, that the longer it will take Feoctist Vsevolodovich to sort the students, the lesser time will remain to play at the end of the lesson. The students already stand in a row in some order, but the teacher has got a call on his smartphone and has left for a while. Sasha can do the following no more than once: choose **any** pair of students (not necessary neighbors) and swap them. Of course, he wants to minimize the number of operations, Feoctist Vsevolodovich will perform while sorting the students in a row.

Input

The first line of the input contains a single integer n ($1 \le n \le 1\,000\,000$) — the number of students attending the current lesson.

The second line contains n distinct integers h_i ($1 \le h_i \le n$). The i-th of these numbers corresponds to the height of the student on the i-th position.

Output

Print two numbers of positions Sasha need to swap in order to minimize the number of Feoctist Vsevolodovich operations. If there are many such pairs, print any of them. If there is no need to swap anyone, print `-1 -1`.

Examples

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

Problem E. Task Manager

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	512 mebibytes

Bomboslav is an intern in Yandex who works on improving the internal task manager. The current version of the manager stores two parameters c_i and u_i for each of n tasks assigned to some employee. These parameters denote importance and urgency of the task, respectively. Higher values correspond to higher importance or urgency.

An employee can choose to perform tasks in any order, with only one condition: if some task i is **both** more important and more urgent than some task j, that is, conditions $c_i > c_j$ and $u_i > u_j$ hold simultaneously, task i should go before task j.

Bomboslav decided to add a tool that will advice employees in what order to perform tasks so that they will not break any condition. That turned out to be too easy, so Bomboslav added one more feature: for every task i, employee could specify the pleasure p_i he would get from performing this task. These values must be distinct for all tasks.

When the values p_i are specified for all n tasks available, the task manager should suggest an order of tasks that does not break any initial conditions, and the values p_i arranged in the same order would form the lexicographically maximum sequence. In other words, from all orders that break no conditions on importance and urgency, the task manager should choose the one where p_i of the first task is the maximum possible. If there is still a tie, the task manager should choose an order that has the maximum possible p_i of the second task, and so on.

Input

The first line of input contains a single integer n, the number of tasks in the manager for some employee ($1 \le n \le 100\,000$).

Then follow n lines with task descriptions. Each of them consists of three non-negative integers c_i, u_i and p_i: importance, urgency and pleasure from this task, respectively ($0 \le c_i, u_i, p_i \le 10^9$). All p_i are guaranteed to be pairwise distinct.

Output

Print a correct order of performing this set of tasks such that the corresponding sequence of p_i is lexicographically the maximum possible. The tasks are numbered from 1 in the order they are given in the input. Each task must occur in this order exactly once. One could easily prove that the answer is always unique.

Examples

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