A. AppendAppendAppend

Time limit: 1 second
Memory limit: 256 megabytes

Momo has a string s which he plays with. After each day, he takes the initial string and appends it to the one that he currently has, that is, if the string is "*abc*", on the first day the string will remain "*abc*", on the second day the string will become "*abcabc*", on the third day the string becomes "*abcabcabc*", and so on.

Bobo has another string t. Bobo is curious whether his string t can be found as a subsequence in Momo's string. He is curious what is the first day when this property will hold.

A string a is a subsequence of a string b if a can be obtained from b by deletion of several (possibly, zero or all) elements.

Input

The first line of the input contains Momo's string s.

The second line of the input contains Bobo's string t.

Both strings have lengths at least 1 and at most $5 \cdot 10^5$ and contain only lowercase letters of the English alphabet.

Output

Output a single integer — the number of the day when Bobo's string will be a subsequence of Momo's string for the first time.

It is guaranteed that for all test cases in this problem such a day exists.

Example

standard input	standard output
dwalkcake cakewalk	2

Note

During the first day, the string is "*dwalkcake*", and doesn't contain "*cakewalk*" as a subsequence.

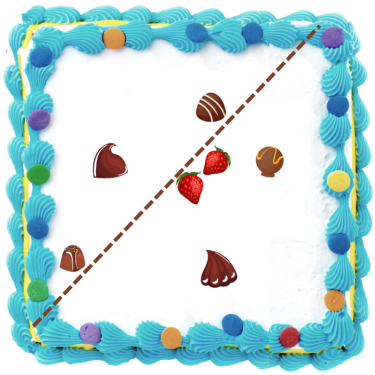
During the second day, the string is "*dwalkcakedwalkcake*", and it contains "*cakewalk*" as a subsequence.

B. Birthday Cake

Time limit: 4 seconds
Memory limit: 256 megabytes

How exciting! Today is your little brother's birthday! That's why you ordered a huge (1×1)-meter cake. It is a special vanilla cake with n sweet chocolate chips and m refreshing strawberries.

You show him your awesome surprise, and... bummer! It turns out that he hates fruit! "*Of course, how could I have forgotten?*" you say. Nonetheless, he has a sweet tooth for chocolate, so he would be happy if you could cut him a piece of the cake that contains no strawberries. To make him happy, you'd want to give him a piece having as many chocolate chips as possible.



The picture above depicts the example test case.

You can only make one cut along a straight line through the cake, and you are not allowed to cut though either chocolate chips or strawberries. What is the maximum number of chocolate chips that you may give your little brother?

Note: The picture above is for illustration purposes. You should consider both chocolate chips and strawberries to be infinitesimally small.

Input

The first line of the input contains two positive integers n $(1 \leq n \leq 50\,000)$ and m $(1 \leq m \leq 100)$ — the number of chocolate chips and strawberries, respectively.

The i-th of the next $n + m$ lines contains two decimal numbers x_i and y_i, $(0 < x_i, y_i < 1)$, representing the coordinates of the i-th ingredient: the first n of the ingredients are chocolate chips, and the remaining m are strawberries.

All numbers are given with at most 6 decimal places. The locations of all $n + m$ ingredients are distinct.

Output

Output a single non-negative integer c, representing the maximum number of chocolate chips that you can give your little brother after cutting the cake exactly once.

Example

standard input	standard output
5 2 0.2 0.6 0.8 0.6 0.6 0.2 0.1 0.2 0.6 0.8 0.6 0.6 0.5 0.5	3

C. COVID

Time limit: 2 seconds
Memory limit: 256 megabytes

During the standard yearly testing, n people are being tested for COVID.

Because of the limited amount of available tests, a procedure called **batch testing** is in place. More specifically, for each of the available m tests, a subset of people is selected and they are tested as a group. If any of the people of the group is positive, the test will yield a positive result (and similarly, if none of the tested people are infected, the test will yield a negative result).

More specifically, test i ($1 \le i \le m$) is done on the group of people with IDs $a_{i,1}, a_{i,2}, \ldots, a_{i,k_i}$. All m tests are perfect (they always yield positive if and only if at least one person of the group is positive), and they, unfortunately, all came back positive.

Before testing, each person was expected to have a 50% probability of being positive. Additionally, all of the probabilities of having COVID were independent (i.e. one person having / not having the disease did not influence the probability of someone else having it). However, given the m positive test results, some people are now more likely to have COVID than others, and it is crucial for the medical team to figure out what is the new status.

Sort the people from least probable to have the disease, to most probable, after the batch testing procedure.

Tip: It can be proven that the posterior probability of person i ($1 \le i \le n$) having the disease is proportional to the number of scenarios (out of the 2^n) in which all m tests output the correct result, and person i is positive.

Input

The first line of the input contains two integers n, m ($1 \le n \le 1000, 1 \le m \le 15$) — the number of people and the number of available tests, respectively.

The following m lines describe the tests. The i-th of them ($1 \le i \le m$) starts with an integer k_i ($1 \le k_i \le n$), the number of people in the test group, followed by k_i numbers between 1 and n, representing the people tested in the i-th test, **in increasing order**.

Output

Output n numbers, representing the IDs of the n people, sorted from least probable to most probable to have COVID. People with equal probability should be sorted in increasing order of IDs.

Examples

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

Note

In the first example, the probabilities of the 5 people are $\frac{8}{11}$, $\frac{7}{11}$, $\frac{6}{11}$, $\frac{6}{11}$, $\frac{1}{2}$.

In the second example, all probabilities are equal to $\frac{4}{7}$.

D. Divisible by 4 Spanning Tree

Time limit: 2 seconds
Memory limit: 256 megabytes

A tree T is **special** if the number of vertices with an odd degree (with respect to T) is a multiple of 4.

You are given a connected graph with n vertices and m edges. Determine if there is a **special** spanning tree in this graph.

As a reminder, a spanning tree of a graph is a subset of the edges of the graph that forms a tree.

Input

The first line of the input contains a single integer t ($1 \le t \le 10\,000$). The description of the test cases follows.

The first line of each test case contains two integers n and m ($2 \le n \le 200\,000$, $n - 1 \le m \le 200\,000$) — the numbers of nodes and edges correspondingly. The i-th of the following m lines contains two integers u_i, v_i ($1 \le u_i, v_i \le n$, $u_i \ne v_i$), indicating that there is an edge between nodes u_i and v_i.

No edge will appear multiple times inside a test case. The sum of n over all test cases won't exceed 200000. The sum of m over all test cases won't exceed 400 000.

Output

For each test case, output if `YES`, if such spanning tree exists, and `NO` otherwise.

Example

standard input	standard output
4 3 2 1 2 2 3 4 3 1 2 1 3 1 4 7 7 1 3 2 3 3 4 4 5 5 6 6 7 7 4 8 8 1 2 2 3 3 4 4 1 1 5 2 6 3 7 4 8	NO YES YES NO

E. Exercise

Time limit: 1 second
Memory limit: 256 megabytes

In a classroom, there are $2n$ students. The i-th student has skill c_i. A teacher is holding an exercise that requires working in pairs. Initially, students $2i-1$ and $2i$ are paired, for each i from 1 to n.

Now, the teacher wants to form new pairs $(a_1, b_1), (a_2, b_2), \ldots, (a_n, b_n)$ with the following conditions:

- Each student is in exactly one pair. That is, each number from 1 to $2n$ appears among numbers $a_1, a_2, \ldots, a_n, b_1, b_2, \ldots, b_n$ exactly once.
- No two students that were in a pair before will be in a pair again. That is, there are no $1 \le i, j \le n$ such that $a_i = 2j-1, b_i = 2j$, or $a_i = 2j, b_i = 2j-1$.

The teacher wants to minimize the sum of differences in skills over the n pairs, as this would be more productive. In other words, $|c_{a_1} - c_{b_1}| + |c_{a_2} - c_{b_2}| + \ldots + |c_{a_n} - c_{b_n}|$ should be as small as possible.

What is the smallest value of $|c_{a_1} - c_{b_1}| + |c_{a_2} - c_{b_2}| + \ldots + |c_{a_n} - c_{b_n}|$ the teacher can achieve?

Input

The first line of the input contains a single integer n ($2 \le n \le 100\,000$).

The second line of the input contains $2n$ integers $c_1, c_2, \ldots, c_{2n}$ ($0 \le c_i \le 10^9$) — the skills of the students.

Output

Output a single integer — the smallest possible value of $|c_{a_1} - c_{b_1}| + |c_{a_2} - c_{b_2}| + \ldots + |c_{a_n} - c_{b_n}|$ that the teacher can achieve.

Examples

standard input	standard output
2 1 2 3 4	4
3 1 9 3 4 2 6	7

Note

In the first example, the teacher can pair up the first student with the third and the second student with the fourth, getting $|3-1| + |4-2| = 4$.

In the second example, the teacher can pair up the first student with the third, the second with the sixth, and the fourth with the fifth, getting $|1-3| + |9-6| + |4-2| = 7$.

F. Fortune over Sportsmanship

Time limit: 2 seconds
Memory limit: 256 megabytes

There is a tennis tournament happening right around the corner, with n vicious participants.

The marketing experts have devised an $n \times n$ matrix P, where $P_{i,j}$ is the popularity score gained if player i competes in a match with player j. They also observed the following social phenomenon: whenever player i competes against player j and wins, player i will inherit all the popularity scores of player j, that is, $P_{i,x}$ becomes the maximum of $P_{i,x}$ and $P_{j,x}$, for all $1 \leq x \leq n$ (and similarly for $P_{x,i}$).

Since fairness isn't a concern, the tournament doesn't have to be perfect. In that sense, any set of $n-1$ matches played in order can be a valid tournament. To top if off, the contestants are numbered from 1 to n in descending order of their performance. In that sense, if players i and j were to compete in a match, where $i < j$, then player i will always win.

Given the popularity matrix P, you have to decide the $n-1$ matches that are going to be played in order during the tournament, such that the total popularity score over the $n-1$ matches is as large as possible. Note that once a match takes place, the loser gets disqualified, therefore she/he cannot not participate in future matches.

Input

The first line of the input contains a single integer n ($1 \leq n \leq 1000$), the number of players.

The i-th of the following n lines contains n integers $P_{i,1}, P_{i,2}, \ldots, P_{i,n}$, describing the i-th row of the popularity matrix P.

It is guaranteed that $1 \leq P_{i,j} \leq 10^6$ and $P_{i,j} = P_{j,i}$ for all $1 \leq i < j \leq n$. Moreover, $P_{i,i} = 0$ for all $1 \leq i \leq n$.

Output

Output n lines. The first line contains a single integer – the maximum possible total popularity score. Each of the following $n-1$ lines should contain two integers, indicating the participants that fight in each of the $n-1$ matches in order. If there are multiple solutions, output any of them.

Example

standard input	standard output
5 0 2 3 4 5 2 0 4 5 6 3 4 0 6 7 4 5 6 0 8 5 6 7 8 0	26 4 5 3 4 2 3 2 1

G. Gears

Time limit: 3 seconds
Memory limit: 256 megabytes

You have an intricate system of n circular gears connected in a line, with their centers fixed in n corresponding axles. The i-th axle is fixed on the wall at coordinate x_i on the (imaginary) number line.

The picture above depicts the example test case.

However, a massive earthquake just hit, and all gears have fallen to the ground, leaving the axles hanging on the wall. Given the radii of the n gears r_i and the coordinates of the axles, you are asked to reinstall the system by putting the gears back onto their original places.

Input

The first line of the input contains a single integer n ($1 \leq n \leq 500\,000$).

The second line of the input contains n integers $x_1, x_2, \ldots, x_n$ ($1 \leq x_i \leq 10^{16}$), the positions of the axles **in increasing order**.

The third line of the input contains n integers $r_1, r_2, \ldots, r_n$ ($1 \leq r_i \leq 10^9$), the radii of the n gears.

Output

Output n numbers $s_1, s_2, \ldots, s_n$ which indicate the correct placement of the gears on their axle. More specifically, s_i should be the radius of the gear that will be placed on the i-th axle from the left. For the placement to be valid, every two neighbouring gears must be touching, and s should be a permutation of r.

It is guaranteed that for all test cases, a solution exists. If multiple solutions exists, you may output any of them.

Example

standard input	standard output
5 2 8 13 22 30 3 5 5 1 4	5 1 4 5 3

H. Hanoi

Time limit: 1 second
Memory limit: 256 megabytes

The original problem "Towers of Hanoi" is about moving n circular disks of **distinct** sizes between 3 rods. In one move, the player can move only the top disk from one rod to the top of another. The disks should always be placed one over the other in decreasing order of sizes. Initially, the n disks reside on rod 1 and the goal is to have them all n reside on rod 3.

There is a myth of an Indian temple where priests are solving the instance of "Towers of Hanoi" with 64 disks since the beginning of time. It is believed that once this instance is solved, the world will end.

However, this "Relaxed Hanoi" problem is not that apocalyptic. In fact, it is pretty optimistic as once you correctly solve it, you will get a positive verdict. In this variation, rod 1 is not subject to the *order rule*. In other words, the disks on rod 1 can reside in any order at any point of time.

For an instance of the "Relaxed Hanoi" problem, provide at most $2 \cdot n^2$ moves that solve the problem.

Input

The first line of the input contains a single integer n ($1 \le n \le 500$) — the number of disks.

The second line of the input contains n integers $p_1, p_2, \ldots, p_n$ — the sizes of the disks on rod 1 from bottom to top (the first is the bottom disk and the n-th is the top disk).

It is guaranteed that $p_1, p_2, \ldots, p_n$ form a permutation (in other words, each number from 1 to n appears exactly once amongst $p_1, p_2, \ldots, p_n$).

Output

The first line of output should contain a single integer k ($0 \le k \le 2 \cdot n^2$) — the number of moves.

Each of the following k lines should contain two integers a_i and b_i ($1 \le a_i, b_i \le 3$), describing a move of the topmost disk from rod a_i to rod b_i.

In the end, all n disks should reside on rod number 3.

Example

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

Note

The provided output has only 9 moves. For $n = 5$, any solution with at most 50 moves will suffice.

- After moves 1,2 and 3, the configuration is $\langle 2, 1 \rangle \langle 4, 3 \rangle \langle 5 \rangle$.
- Move 4 is legal as the first rod is relaxed. The configuration is $\langle 2, 1, 3 \rangle \langle 4 \rangle \langle 5 \rangle$.
- After moves 5 and 6, the configuration is $\langle 2, 1 \rangle \langle \rangle \langle 5, 4, 3 \rangle$.
- Last moves 7, 8 and 9 determine the final configuration is $\langle \rangle \langle \rangle \langle 5, 4, 3, 2, 1 \rangle$.

I. Inadequate Operation

Time limit: 1 second
Memory limit: 256 megabytes

You are given an array a of n nonnegative integers. In one operation, you can choose any i such that $1 \le i \le n-1$ and $\max(a_i, a_{i+1}) > 0$, and replace both a_i and a_{i+1} by $\max(a_i, a_{i+1}) - 1$.

Find the smallest number of operations required to make all elements equal to 0. It can be proven that it's always possible to make all numbers equal to 0.

Input

The first line of the input contains a single integer n ($2 \le n \le 200\,000$) — the number of integers.

The second line of the input contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 10^9$) — the elements of the array.

Output

Output a single integer — the smallest number of operations required to make all elements equal to 0.

Examples

standard input	standard output
3 2 0 22	24
3 0 2022 0	2022
6 30 20 10 10 20 30	70

Note

In the first test case, you can apply this operation 2 times to $i = 1$, and then 22 times to $i = 2$.

In the second test case, you can apply this operation 2022 times to $i = 1$.

J. Joyful Death

Time limit: 3 seconds
Memory limit: 256 megabytes

You are Psycho Cookmaster, a devil-possessed spirit with a surprisingly humble love for food. Tonight is the opening night of your brand new restaurant – "Food to Die For" – and it's already full with reservations from n elves! The i-th elf has a current sickness of s_i.

The n elves will come in order tonight for their unforgettable dinner. To avoid having surprises, you have prepared m dishes in advance, each of them having a health index h_i and a tastiness index of t_i. If an elf x eats a dish y such that $h_y < s_x$, then elf x will die.

Tonight is a great opportunity to show your evil side; after all, you are a psychopath god! Consequently, you want to distribute the m dishes such that the elves that eat them will die, but you also want to show them some remorse and your love for tasty feasts. Therefore, you decided to give the elves that are going to die delicious food so that they will die with no regret. The elves that wouldn't die will have their reservations cancelled, and receive a nifty coupon to spend some other day at your restaurant.

In other words, your goal is to distribute some of the dishes amongst some of the elves (possibly none), such that the sum of tastiness of the dishes eaten by the elves that are going to die is as large as possible.

The only issue is that you do not know when the so-called 'good gods' might interfere with your plan. Therefore, knowing that the n elves will come to the restaurant in order, you have to solve this problem independently for the first i elves, for all $1 \leq i \leq n$.

Input

The first line of the input contains two positive integers n and m ($1 \leq n, m \leq 200\,000$) – the number of elves and the number of dishes, respectively.

The second line of the input contains n positive integers $s_1, s_2, \ldots, s_n$ ($1 \leq s_i \leq 10^9$) — the sickness of each elf.

Finally, the i-th of the next m lines contains two integers h_i and t_i ($1 \leq h_i, t_i \leq 10^9$) — the health index and the tastiness index of dish i.

Output

Output n integers, the i-th of them should be the answer for the first i elves.

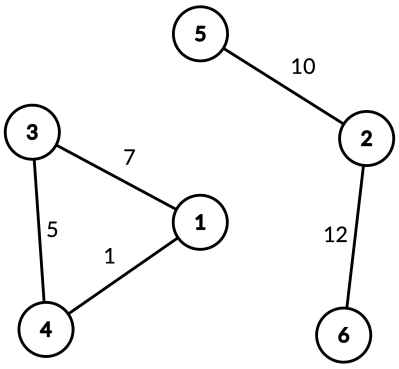
Example

standard input	standard output
5 3 8 10 5 1 6 6 10 5 12 9 4	12 22 22 22 26

K. Knowledge Testing Problem

Time limit: 3 seconds
Memory limit: 512 megabytes

Every respectable contest must have a textbook graph problem. No convoluted processes, no weird observations; just pure, raw algorithmics. Lucky for you, you've just found it!



The picture above depicts the example test case.

You are given an undirected, weighted graph with n vertices and m edges, as well as q queries of form (a_i, b_i). For each of the queries, find the length of the shortest path between vertices a_i and b_i.

Input

The input consists of $m + q + 1$ lines. The first line of the input will contain three integers n, m, and q ($1 \leq n \leq 100\,000$, $1 \leq m \leq 200\,000$, $1 \leq q \leq 25\,000$). Each of the next m lines contain three integers u_i, v_i, and w_i, denoting undirected edge between vertices u_i and v_i of length w_i ($1 \leq u_i, v_i \leq n, 1 \leq w_i \leq 10^9$). Finally, each of the last q lines contain two integers a_i, b_i ($1 \leq a_i, b_i \leq n$).

There is at most one edge between any pair of vertices, and no edges that connect a vertex with itself.

Moreover, it is guaranteed that all m edges u_i, v_i satisfy $|u_i - v_i| \leq 10$.

Output

Output q lines. The i-th line should contain a single positive integer, representing the minimum length of a path that connects the vertices in the i-th query. If there is no such path, output -1 for that specific query.

Example

standard input	standard output
6 5 3 1 3 7 3 4 5 1 4 1 2 5 10 2 6 12 6 5 1 3 1 5	22 6 -1

L. Level Up

Time limit: 2 seconds
Memory limit: 256 megabytes

Steve discovered a new RPG video game he really wants to play. This game is unique: players who can finish it optimally get their money back for the game! What a world for Steve to live in!

The game has n realms. The player starts in realm 1, with level 1 and a chosen quantity of hit points.

Each realm has a number of mobs, each with a specific strength. In each realm, the player should choose to engage and fight **one non-empty subset** of the mobs present in that realm. The fight happens in a turn-based combat system. First, all alive monsters attack, each of them yielding a damage to the player equal to its strength. Then, the player chooses one of the engaged mobs and kills it (that mob won't attack the player ever again). Every kill levels up the player (up to level m).

Once all the engaged mobs in realm i are dead, the player heals h_l hit points (where l is its level after killing all the mobs) and advances to realm $i+1$. Advancing from realm n will **finish the game**. There is no limit on the number of hit points the player can have in the play-through.

Steve's goal is to **finish the game at the maximum level** m without ever having 0 or less hit points on its play-through. Steve must play optimally, meaning that he needs to choose the minimum starting quantity of hit points possible such that he can **finish the game with max-level**.

Input

The first line of the input contains two integers n ($1 \le n \le 100$) and m ($1 \le m \le 50\,000$) — the number of realms, and the maximum level achievable in the game, respectively.

The second line of the input contains m integers $h_1, h_2, \ldots, h_m$ ($1 \le h_1 \le h_2 \le \ldots \le h_m \le 10^6$) — the number of hit points the player gets healed, based on its level.

Finally, the next n lines describe the n realms. The i-th of these lines contains a number k_i ($1 \le k_i \le 2000$) of mobs in realm i, followed by k_i integers $s_{i,1}, s_{i,2}, \ldots, s_{i,k_i}$ — the strengths of each of the k_i mobs.

All strengths are between 1 and 10^4 inclusively. Moreover, it is guaranteed that there are at least $m-1$ mobs in the entire game.

Output

Output a single positive integer — the minimum starting health possible.

Example

standard input	standard output
3 4 1 2 10 11 3 6 2 3 2 11 12 4 9 11 10 5	9

Note

Steve enters the first realm with 9 HP. He engages the mobs with 2 and 3 strength. He starts the fight by taking $2+3=5$ damage, then kills the mob with 3 strength, takes another 2 damage, and kills the remaining mob. It gains level 3 in the process, gets healed $h_3 = 10$ HP and advances with 12 total HP.

In the second realm, Steve engages the mob with 11 strength, takes 11 damage, and kills it. He is now level 4, therefore he gets healed $h_4 = 11$ HP and advances with 12 total HP.

In the third realm, Steve engages one of the mobs (no matter which one), takes damage, kills it (without levelling up), and finishes the game at the maximum level 4.

M. Mousetrap

Time limit: 3 seconds
Memory limit: 256 megabytes

Jerry has a network of n chambers connected by $n-1$ bi-directional tubes, such that any two chambers are (directly or indirectly) connected. In each of the chambers is a certain quantity of cheese. In chamber i ($1 \leq i \leq n$) the cheese quantity is c_i. The network has an exit at chamber n.

A mouse is initially located in chamber 1. At each step, the mouse will use its smell to detect the quantity of cheese in the neighbouring chambers, and will randomly choose to go through a tube with a likelihood proportional to the quantity of cheese in the corresponding chamber. The mouse will never consider going back to a previous chamber. If there's no next tube to choose, the mouse will get trapped.

Jerry wants to maximize the probability of the mouse to reach the exit. In order to do that, he may add cheese to any of the chambers. However, he can only add integral amounts of cheese to each chamber, and the total quantity must not exceed x.

How much cheese should Jerry add to each chamber, so that he maximizes the chances of the mouse escaping?

Input

The first line of the input contains the numbers n ($2 \leq n \leq 200\,000$) and x ($1 \leq x \leq 10^9$).

The second line of the input contains n integers $c_1, c_2, \ldots, c_n$ ($1 \leq c_i \leq 10^9$).

The i-th of the next $n-1$ lines contains two integers u_i and v_i ($1 \leq u_i, v_i \leq n, u_i \neq v_i$), indicating that chambers u_i and v_i are directly connected by a tube.

It is guaranteed that the network of chambers is connected. Moreover, it is guaranteed that the total quantity of cheese initially in the chambers is at most 10^9.

Output

Output n non-negative integers. The i-th integer represents the quantity of cheese that Jerry should add to the i-th chamber. If there are multiple solutions, you may output any of them.

Examples

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

N. Nusret Gökçe

Time limit: 1 second
Memory limit: 256 megabytes

Nusret Gökçe is, by all means, one of the most valuable chefs to this date (at least, judging by the bills his customers have to pay). He is preparing his signature dish: the **24-karat gold steak** (literally). He cooked the perfect medium rare steak with a shiny crust, cut it tenderly into n pieces, and was about to sprinkle just a pinch of salt, when the unthinkable happened: his novice apprentice tripped next to the steak, spreading salt wildly throughout the steak pieces! Now, the otherwise perfect gold steak has $s_1, s_2, \ldots, s_n$ grains of salt, respectively on each of the slices.

Nusret nourishing his signature dish. Source: eater.com

Nusret isn't the one to throw away dishes (especially when they are worth their weight in gold); therefore, he decided to fix the steak instead. He cannot remove the existing salt without compromising the texture of the steak; instead, he will use his salt sprinkling skills and grain-level precision to add more salt to some of the n slices in order to make the salt distribution more uniform.

He knows that the customer will eat the n slices in order, and so they will not sense anything suspicious if the amount of salt between every two adjacent slices does not differ by more than m grains. At the same time, he doesn't want to make the steak unnecessarily salty; therefore, he will sprinkle the salt in such a way that the total number of grains of salt is as small as possible.

How many grains of salt will be on each slice after Nusret "fixes" his signature dish?

Input

The first line of the input contains two integers n ($1 \leq n \leq 100\,000$) and m ($0 \leq m \leq 10^9$) — the number of slices in Nusret's signature dish, and the maximum allowed difference in saltiness.

The second line of the input contains n integers $s_1, s_2, \ldots, s_n$ ($1 \leq s_i \leq 10^9$) — the number of grains of salt on each of the n slices.

Output

Output n integers $t_1, t_2, \ldots, t_n$, representing the number of grains of salt on each of the n slices after Nusret fixes the dish.

Any solution in which the total number of grains of salt is as small as possible will be accepted.

Examples

standard input	standard output
8 3 1 16 4 2 3 8 1 9	13 16 13 10 7 8 6 9
5 0 1 5 7 1 6	7 7 7 7 7