Educational Codeforces Round 95 (Rated for Div. 2)

A. Buying Torches

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

You are playing a very popular game called Cubecraft. Initially, you have one stick and want to craft k torches. One torch can be crafted using **one stick and one coal**.

Hopefully, you've met a very handsome wandering trader who has two trade offers:

- exchange 1 stick for x sticks (you lose 1 stick and gain x sticks).
- exchange y sticks for 1 coal (you lose y sticks and gain 1 coal).

During one trade, you can use **only one** of these two trade offers. You can use each trade offer any number of times you want to, in any order.

Your task is to find the minimum number of trades you need to craft at least k torches. The answer always exists under the given constraints.

You have to answer t independent test cases.

Input

The first line of the input contains one integer t ($1 \le t \le 2 \cdot 10^4$) — the number of test cases. Then t test cases follow.

The only line of the test case contains three integers x, y and k ($2 \le x \le 10^9$; $1 \le y, k \le 10^9$) — the number of sticks you can buy with one stick, the number of sticks required to buy one coal and the number of torches you need, respectively.

Output

For each test case, print the answer: the minimum number of trades you need to craft at least k torches. The answer always exists under the given constraints.

Example

input
5 2 1 5 42 13 24 12 11 12 1000000000 1000000000 1000000000 2 1000000000 1000000000
output
14 33 25 2000000003 1000000001999999999

B. Negative Prefixes

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

You are given an array a, consisting of n integers.

Each position i ($1 \le i \le n$) of the array is either locked or unlocked. You can take the values on the unlocked positions, rearrange them in any order and place them back into the unlocked positions. You are not allowed to remove any values, add the new ones or rearrange the values on the locked positions. You are allowed to leave the values in the same order as they were.

For example, let $a = [-1, 1, \underline{3}, 2, \underline{-2}, 1, -4, \underline{0}]$, the underlined positions are locked. You can obtain the following arrays:

- $[-1, 1, \underline{3}, 2, \underline{-2}, 1, -4, \underline{0}]$;
- $[-4, -1, \underline{3}, 2, \underline{-2}, 1, 1, \underline{0}]$;
- $[1, -1, \underline{3}, 2, \underline{-2}, 1, -4, \underline{0}]$;
- $[1, 2, \underline{3}, -1, \underline{-2}, -4, 1, \underline{0}]$;

- and some others.

Let p be a sequence of prefix sums of the array a after the rearrangement. So $p_1 = a_1$, $p_2 = a_1 + a_2$, $p_3 = a_1 + a_2 + a_3$, $\ldots$, $p_n = a_1 + a_2 + \cdots + a_n$.

Let k be the maximum j ($1 \le j \le n$) such that $p_j < 0$. If there are no j such that $p_j < 0$, then $k = 0$.

Your goal is to rearrange the values in such a way that k is minimum possible.

Output the array a after the rearrangement such that the value k for it is minimum possible. If there are multiple answers then print any of them.

Input
The first line contains a single integer t ($1 \le t \le 1000$) — the number of testcases.

Then t testcases follow.

The first line of each testcase contains a single integer n ($1 \le n \le 100$) — the number of elements in the array a.

The second line of each testcase contains n integers $a_1, a_2, \ldots, a_n$ ($-10^5 \le a_i \le 10^5$) — the initial array a.

The third line of each testcase contains n integers $l_1, l_2, \ldots, l_n$ ($0 \le l_i \le 1$), where $l_i = 0$ means that the position i is unlocked and $l_i = 1$ means that the position i is locked.

Output
Print n integers — the array a after the rearrangement. Value k (the maximum j such that $p_j < 0$ (or 0 if there are no such j)) should be minimum possible. For each locked position the printed value should be equal to the initial one. The values on the unlocked positions should be an arrangement of the initial ones.

If there are multiple answers then print any of them.

Example

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

Note
In the first testcase you can rearrange all values however you want but any arrangement will result in $k = 0$. For example, for an arrangement $[1, 2, 3]$, $p = [1, 3, 6]$, so there are no j such that $p_j < 0$. Thus, $k = 0$.

In the second testcase you are not allowed to rearrange any elements. Thus, the printed array should be exactly the same as the initial one.

In the third testcase the prefix sums for the printed array are $p = [-8, -14, -13, -9, -5, 2, 0]$. The maximum j is 5, thus $k = 5$. There are no arrangements such that $k < 5$.

In the fourth testcase $p = [-4, -4, -3, 3, 6]$.

In the fifth testcase $p = [-1, 3, 10, 2, 12, 11]$.

C. Mortal Kombat Tower

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

You and your friend are playing the game Mortal Kombat XI. You are trying to pass a challenge tower. There are n bosses in this tower, numbered from 1 to n. The type of the i-th boss is a_i. If the i-th boss is easy then its type is $a_i = 0$, otherwise this boss is hard and its type is $a_i = 1$.

During one session, either you or your friend can kill **one or two** bosses (neither you nor your friend can skip the session, so the minimum number of bosses killed during one session is at least one). After your friend session, your session begins, then again your friend session begins, your session begins, and so on. **The first session is your friend's session**.

Your friend needs to get good because he can't actually kill hard bosses. To kill them, he uses skip points. One skip point can be used to kill one hard boss.

Your task is to find the **minimum** number of skip points your friend needs to use so you and your friend kill all n bosses in the given order.

For example: suppose $n = 8$, $a = [1, 0, 1, 1, 0, 1, 1, 1]$. Then the best course of action is the following:

- your friend kills two first bosses, using one skip point for the first boss;
- you kill the third and the fourth bosses;
- your friend kills the fifth boss;
- you kill the sixth and the seventh bosses;
- your friend kills the last boss, using one skip point, so the tower is completed using two skip points.

You have to answer t independent test cases.

Input

The first line of the input contains one integer t ($1 \le t \le 2 \cdot 10^4$) — the number of test cases. Then t test cases follow.

The first line of the test case contains one integer n ($1 \le n \le 2 \cdot 10^5$) — the number of bosses. The second line of the test case contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 1$), where a_i is the type of the i-th boss.

It is guaranteed that the sum of n does not exceed $2 \cdot 10^5$ ($\sum n \le 2 \cdot 10^5$).

Output

For each test case, print the answer: the **minimum** number of skip points your friend needs to use so you and your friend kill all n bosses in the given order.

Example

input
6 8 1 0 1 1 0 1 1 1 5 1 1 1 1 0 7 1 1 1 1 0 0 1 6 1 1 1 1 1 1 1 1 1 0

output
2 2 2 2 1 0

D. Trash Problem

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Vova decided to clean his room. The room can be represented as the coordinate axis OX. There are n piles of trash in the room, coordinate of the i-th pile is the integer p_i. All piles have **different** coordinates.

Let's define a *total cleanup* as the following process. The goal of this process is to collect **all** the piles in **no more than two** different x coordinates. To achieve this goal, Vova can do several (possibly, zero) moves. During one move, he can choose some x and move **all piles** from x to $x + 1$ or $x - 1$ using his broom. Note that he can't choose how many piles he will move.

Also, there are two types of queries:

- $0\ x$ — remove a pile of trash from the coordinate x. It is guaranteed that there is a pile in the coordinate x at this moment.
- $1\ x$ — add a pile of trash to the coordinate x. It is guaranteed that there is no pile in the coordinate x at this moment.

Note that it is possible that there are zero piles of trash in the room at some moment.

Vova wants to know the **minimum** number of moves he can spend if he wants to do a *total cleanup* before any queries. He also wants to know this number of moves after applying each query. Queries are applied in the given order. Note that the *total cleanup*

doesn't actually happen and doesn't change the state of piles. It is only used to calculate the number of moves.

For better understanding, please read the **Notes** section below to see an explanation for the first example.

Input

The first line of the input contains two integers n and q ($1 \le n, q \le 10^5$) — the number of piles in the room before all queries and the number of queries, respectively.

The second line of the input contains n distinct integers $p_1, p_2, \ldots, p_n$ ($1 \le p_i \le 10^9$), where p_i is the coordinate of the i-th pile.

The next q lines describe queries. The i-th query is described with two integers t_i and x_i ($0 \le t_i \le 1; 1 \le x_i \le 10^9$), where t_i is 0 if you need to remove a pile from the coordinate x_i and is 1 if you need to add a pile to the coordinate x_i. It is guaranteed that for $t_i = 0$ there is such pile in the current set of piles and for $t_i = 1$ there is no such pile in the current set of piles.

Output

Print $q + 1$ integers: the minimum number of moves Vova needs to do a *total cleanup* before the first query and after each of q queries.

Examples

input
5 6 1 2 6 8 10 1 4 1 9 0 6 0 10 1 100 1 50
output
5 7 7 5 4 8 49

input
5 8 5 1 2 4 3 0 1 0 2 0 3 0 4 0 5 1 1000000000 1 1 1 500000000
output
3 2 1 0 0 0 0 0 499999999

Note

Consider the first example.

Initially, the set of piles is $[1, 2, 6, 8, 10]$. The answer before the first query is 5 because you can move all piles from 1 to 2 with one move, all piles from 10 to 8 with 2 moves and all piles from 6 to 8 with 2 moves.

After the first query, the set becomes $[1, 2, 4, 6, 8, 10]$. Then the answer is 7 because you can move all piles from 6 to 4 with 2 moves, all piles from 4 to 2 with 2 moves, all piles from 2 to 1 with 1 move and all piles from 10 to 8 with 2 moves.

After the second query, the set of piles becomes $[1, 2, 4, 6, 8, 9, 10]$ and the answer is the same (and the previous sequence of moves can be applied to the current set of piles).

After the third query, the set of piles becomes $[1, 2, 4, 8, 9, 10]$ and the answer is 5 because you can move all piles from 1 to 2 with 1 move, all piles from 2 to 4 with 2 moves, all piles from 10 to 9 with 1 move and all piles from 9 to 8 with 1 move.

After the fourth query, the set becomes $[1, 2, 4, 8, 9]$ and the answer is almost the same (the previous sequence of moves can be applied without moving piles from 10).

After the fifth query, the set becomes $[1, 2, 4, 8, 9, 100]$. You can move all piles from 1 and further to 9 and keep 100 at its place. So the answer is 8.

After the sixth query, the set becomes $[1, 2, 4, 8, 9, 50, 100]$. The answer is 49 and can be obtained with almost the same sequence of moves as after the previous query. The only difference is that you need to move all piles from 50 to 9 too.

E. Expected Damage

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

You are playing a computer game. In this game, you have to fight n monsters.

To defend from monsters, you need a shield. Each shield has two parameters: its current durability a and its defence rating b. Each monster has only one parameter: its strength d.

When you fight a monster with strength d while having a shield with current durability a and defence b, there are three possible outcomes:

- if $a = 0$, then you receive d damage;
- if $a > 0$ and $d \geq b$, you receive no damage, but the current durability of the shield decreases by 1;
- if $a > 0$ and $d < b$, nothing happens.

The i-th monster has strength d_i, and you will fight each of the monsters exactly once, in some random order (all $n!$ orders are equiprobable). You have to consider m different shields, the i-th shield has initial durability a_i and defence rating b_i. For each shield, calculate the expected amount of damage you will receive if you take this shield and fight the given n monsters in random order.

Input

The first line contains two integers n and m ($1 \leq n, m \leq 2 \cdot 10^5$) — the number of monsters and the number of shields, respectively.

The second line contains n integers d_1, d_2, ..., d_n ($1 \leq d_i \leq 10^9$), where d_i is the strength of the i-th monster.

Then m lines follow, the i-th of them contains two integers a_i and b_i ($1 \leq a_i \leq n$; $1 \leq b_i \leq 10^9$) — the description of the i-th shield.

Output

Print m integers, where the i-th integer represents the expected damage you receive with the i-th shield as follows: it can be proven that, for each shield, the expected damage is an irreducible fraction $\frac{x}{y}$, where y is coprime with 998244353. You have to print the value of $x \cdot y^{-1} \bmod 998244353$, where y^{-1} is the inverse element for y ($y \cdot y^{-1} \bmod 998244353 = 1$).

Examples

input
3 2 1 3 1 2 1 1 2
output
665496237 1

input
3 3 4 2 6 3 1 1 2 2 3
output
0 8 665496236

F. Equal Product

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

You are given four integers n, m, l and r.

Let's name a tuple (x_1, y_1, x_2, y_2) as *good* if:

1. $1 \leq x_1 < x_2 \leq n$;

2. $1 \le y_2 < y_1 \le m$;
3. $x_1 \cdot y_1 = x_2 \cdot y_2$;
4. $l \le x_1 \cdot y_1 \le r$.

Find any good tuple **for each** x_1 **from** 1 **to** n **inclusive**.

Input
The first line contains two integers n and m ($1 \le n, m \le 2 \cdot 10^5$).

The second line contains two integers l and r ($1 \le l \le r \le nm$).

Output
For each x_1 from 1 to n inclusive:

- if there are no such four integers, print -1;
- otherwise, print four integers x_1, y_1, x_2 and y_2. If there are multiple answers, print any of them.

Examples

input
8 20 91 100
output
-1 -1 -1 -1 -1 6 16 8 12 -1 -1

input
4 5 1 10
output
1 2 2 1 2 3 3 2 -1 -1

input
5 12 16 60
output
-1 2 9 3 6 3 8 4 6 4 5 5 4 -1

G. Three Occurrences

time limit per test: 5 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

You are given an array a consisting of n integers. We denote the subarray $a[l..r]$ as the array $[a_l, a_{l+1}, \dots, a_r]$ ($1 \le l \le r \le n$).

A subarray is considered *good* if every integer that occurs in this subarray occurs there **exactly thrice**. For example, the array $[1, 2, 2, 2, 1, 1, 2, 2, 2]$ has three good subarrays:

- $a[1..6] = [1, 2, 2, 2, 1, 1]$;
- $a[2..4] = [2, 2, 2]$;
- $a[7..9] = [2, 2, 2]$.

Calculate the number of good subarrays of the given array a.

Input
The first line contains one integer n ($1 \le n \le 5 \cdot 10^5$).

The second line contains n integers a_1, a_2, ..., a_n ($1 \le a_i \le n$).

Output
Print one integer — the number of good subarrays of the array a.

Examples

input
9 1 2 2 2 1 1 2 2 2
output
3

input
10 1 2 3 4 1 2 3 1 2 3
output
0

input
12 1 2 3 4 3 4 2 1 3 4 2 1
output
1