[Juniors] - Greedy #4

A. I Wanna Be the Guy

1 second, 256 megabytes

There is a game called "I Wanna Be the Guy", consisting of n levels. Little X and his friend Little Y are addicted to the game. Each of them wants to pass the whole game.

Little X can pass only p levels of the game. And Little Y can pass only q levels of the game. You are given the indices of levels Little X can pass and the indices of levels Little Y can pass. Will Little X and Little Y pass the whole game, if they cooperate each other?

Input

The first line contains a single integer n ($1 \le n \le 100$).

The next line contains an integer p ($0 \le p \le n$) at first, then follows p distinct integers $a_1, a_2, ..., a_p$ ($1 \le a_i \le n$). These integers denote the indices of levels Little X can pass. The next line contains the levels Little Y can pass in the same format. It's assumed that levels are numbered from 1 to n.

Output

If they can pass all the levels, print "`I become the guy.`". If it's impossible, print "`Oh, my keyboard!`" (without the quotes).

input

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

output

```
I become the guy.
```

input

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

output

```
Oh, my keyboard!
```

In the first sample, Little X can pass levels [1 2 3], and Little Y can pass level [2 4], so they can pass all the levels both.

In the second sample, no one can pass level 4.

B. Two Arrays And Swaps

1 second, 256 megabytes

You are given two arrays a and b both consisting of n positive (greater than zero) integers. You are also given an integer k.

In one move, you can choose two indices i and j ($1 \le i, j \le n$) and swap a_i and b_j (i.e. a_i becomes b_j and vice versa). Note that i and j can be equal or different (in particular, swap a_2 with b_2 or swap a_3 and b_9 both are acceptable moves).

Your task is to find the **maximum** possible sum you can obtain in the array a if you can do no more than (i.e. at most) k such moves (swaps).

You have to answer t independent test cases.

Input

The first line of the input contains one integer t ($1 \le t \le 200$) — the number of test cases. Then t test cases follow.

The first line of the test case contains two integers n and k ($1 \le n \le 30; 0 \le k \le n$) — the number of elements in a and b and the maximum number of moves you can do. The second line of the test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 30$), where a_i is the i-th element of a. The third line of the test case contains n integers $b_1, b_2, \ldots, b_n$ ($1 \le b_i \le 30$), where b_i is the i-th element of b.

Output

For each test case, print the answer — the **maximum** possible sum you can obtain in the array a if you can do no more than (i.e. at most) k swaps.

input

```
5
2 1
1 2
3 4
5 5
5 5 6 6 5
1 2 5 4 3
5 3
1 2 3 4 5
10 9 10 10 9
4 0
2 2 4 3
2 4 2 3
4 4
1 2 2 1
4 4 5 4
```

output

```
6
27
39
11
17
```

In the first test case of the example, you can swap $a_1 = 1$ and $b_2 = 4$, so $a = [4, 2]$ and $b = [3, 1]$.

In the second test case of the example, you don't need to swap anything.

In the third test case of the example, you can swap $a_1 = 1$ and $b_1 = 10$, $a_3 = 3$ and $b_3 = 10$ and $a_2 = 2$ and $b_4 = 10$, so $a = [10, 10, 10, 4, 5]$ and $b = [1, 9, 3, 2, 9]$.

In the fourth test case of the example, you cannot swap anything.

In the fifth test case of the example, you can swap arrays a and b, so $a = [4, 4, 5, 4]$ and $b = [1, 2, 2, 1]$.

C. Yet Another Two Integers Problem

1 second, 256 megabytes

You are given two integers a and b.

In one move, you can choose some **integer** k from 1 to 10 and add it to a or subtract it from a. In other words, you choose an integer $k \in [1; 10]$ and perform $a := a + k$ or $a := a - k$. You may use **different** values of k in different moves.

Your task is to find the **minimum** number of moves required to obtain b from a.

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 two integers a and b ($1 \le a, b \le 10^9$).

Output

For each test case, print the answer: the minimum number of moves required to obtain b from a.

input

```
6
5 5
13 42
18 4
1337 420
123456789 1000000000
100500 9000
```

output

```
0
3
2
92
87654322
9150
```

In the first test case of the example, you don't need to do anything.

In the second test case of the example, the following sequence of moves can be applied: $13 \to 23 \to 32 \to 42$ (add 10, add 9, add 10).

In the third test case of the example, the following sequence of moves can be applied: $18 \to 10 \to 4$ (subtract 8, subtract 6).

D. Even Array

2 seconds, 256 megabytes

You are given an array $a[0 \ldots n-1]$ of length n which consists of non-negative integers. **Note that array indices start from zero.**

An array is called *good* if the parity of each index matches the parity of the element at that index. More formally, an array is good if for all i $(0 \le i \le n-1)$ the equality $i \bmod 2 = a[i] \bmod 2$ holds, where $x \bmod 2$ is the remainder of dividing x by 2.

For example, the arrays $[0, 5, 2, 1]$ and $[0, 17, 0, 3]$ are good, and the array $[2, 4, 6, 7]$ is bad, because for $i = 1$, the parities of i and $a[i]$ are different: $i \bmod 2 = 1 \bmod 2 = 1$, but $a[i] \bmod 2 = 4 \bmod 2 = 0$.

In one move, you can take **any** two elements of the array and swap them (these elements are not necessarily adjacent).

Find the minimum number of moves in which you can make the array a good, or say that this is not possible.

Input

The first line contains a single integer t $(1 \le t \le 1000)$ — the number of test cases in the test. Then t test cases follow.

Each test case starts with a line containing an integer n $(1 \le n \le 40)$ — the length of the array a.

The next line contains n integers $a_0, a_1, \ldots, a_{n-1}$ $(0 \le a_i \le 1000)$ — the initial array.

Output

For each test case, output a single integer — the minimum number of moves to make the given array a good, or -1 if this is not possible.

input

```
4
4
3 2 7 6
3
3 2 6
1
7
7
4 9 2 1 18 3 0
```

output

```
2
1
-1
0
```

In the first test case, in the first move, you can swap the elements with indices 0 and 1, and in the second move, you can swap the elements with indices 2 and 3.

In the second test case, in the first move, you need to swap the elements with indices 0 and 1.

In the third test case, you cannot make the array good.

E. Honest Coach

2 seconds, 256 megabytes

There are n athletes in front of you. Athletes are numbered from 1 to n from left to right. You know the strength of each athlete — the athlete number i has the strength s_i.

You want to split all athletes into two teams. Each team must have at least one athlete, and each athlete must be exactly in one team.

You want the strongest athlete from the first team to differ as little as possible from the weakest athlete from the second team. Formally, you want to split the athletes into two teams A and B so that the value $|\max(A) - \min(B)|$ is as small as possible, where $\max(A)$ is the maximum strength of an athlete from team A, and $\min(B)$ is the minimum strength of an athlete from team B.

For example, if $n = 5$ and the strength of the athletes is $s = [3, 1, 2, 6, 4]$, then one of the possible split into teams is:

- first team: $A = [1, 2, 4]$,
- second team: $B = [3, 6]$.

In this case, the value $|\max(A) - \min(B)|$ will be equal to $|4 - 3| = 1$. This example illustrates one of the ways of optimal split into two teams.

Print the minimum value $|\max(A) - \min(B)|$.

Input

The first line contains an integer t $(1 \le t \le 1000)$ — the number of test cases in the input. Then t test cases follow.

Each test case consists of two lines.

The first line contains positive integer n $(2 \le n \le 50)$ — number of athletes.

The second line contains n positive integers $s_1, s_2, \ldots, s_n$ $(1 \le s_i \le 1000)$, where s_i — is the strength of the i-th athlete. Please note that s values may not be distinct.

Output

For each test case print one integer — the minimum value of $|\max(A) - \min(B)|$ with the optimal split of all athletes into two teams. Each of the athletes must be a member of exactly one of the two teams.

input

```
5
5
3 1 2 6 4
6
2 1 3 2 4 3
4
7 9 3 1
2
1 1000
3
100 150 200
```

output

```
1
0
2
999
50
```

The first test case was explained in the statement. In the second test case, one of the optimal splits is $A = [2, 1]$, $B = [3, 2, 4, 3]$, so the answer is $|2 - 2| = 0$.

F. Minimal Square

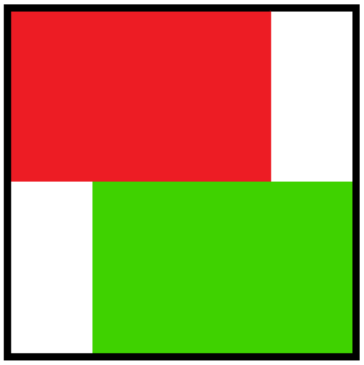
2 seconds, 256 megabytes

Find the minimum area of a **square** land on which you can place two identical rectangular $a \times b$ houses. The sides of the houses should be parallel to the sides of the desired square land.

Formally,

- You are given two identical rectangles with side lengths a and b ($1 \le a, b \le 100$) — positive integers (you are given just the sizes, but **not** their positions).
- Find the square of the minimum area that contains both given rectangles. Rectangles can be rotated (both or just one), moved, but the sides of the rectangles should be parallel to the sides of the desired square.

Two rectangles can touch each other (side or corner), but cannot intersect. Rectangles can also touch the sides of the square but must be completely inside it. You can rotate the rectangles. Take a look at the examples for a better understanding.



The picture shows a square that contains red and green rectangles.

Input

The first line contains an integer t ($1 \le t \le 10\,000$) —the number of test cases in the input. Then t test cases follow.

Each test case is a line containing two integers a, b ($1 \le a, b \le 100$) — side lengths of the rectangles.

Output

Print t answers to the test cases. Each answer must be a single integer — minimal area of square land, that contains two rectangles with dimensions $a \times b$.

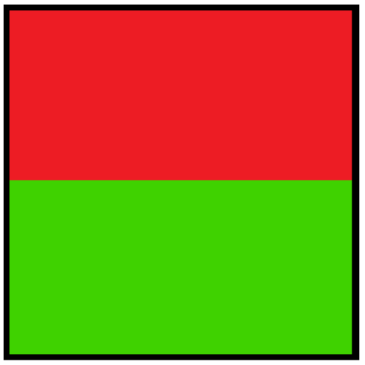
input

```
8
3 2
4 2
1 1
3 1
4 7
1 3
7 4
100 100
```

output

```
16
16
4
9
64
9
64
40000
```

Below are the answers for the first two test cases:



G. Restore the Permutation by Merger

1 second, 256 megabytes

A permutation of length n is a sequence of integers from 1 to n of length n containing each number exactly once. For example, $[1]$, $[4, 3, 5, 1, 2]$, $[3, 2, 1]$ are permutations, and $[1, 1]$, $[0, 1]$, $[2, 2, 1, 4]$ are not.

There was a permutation $p[1 \dots n]$. It was merged with itself. In other words, let's take two instances of p and insert elements of the second p into the first maintaining relative order of elements. The result is a sequence of the length $2n$.

For example, if $p = [3, 1, 2]$ some possible results are: $[3, 1, 2, 3, 1, 2]$, $[3, 3, 1, 1, 2, 2]$, $[3, 1, 3, 1, 2, 2]$. The following sequences are not possible results of a merging: $[1, 3, 2, 1, 2, 3]$, $[3, 1, 2, 3, 2, 1]$, $[3, 3, 1, 2, 2, 1]$.

For example, if $p = [2, 1]$ the possible results are: $[2, 2, 1, 1]$, $[2, 1, 2, 1]$. The following sequences are not possible results of a merging: $[1, 1, 2, 2]$, $[2, 1, 1, 2]$, $[1, 2, 2, 1]$.

Your task is to restore the permutation p by the given resulting sequence a. It is guaranteed that the answer **exists and is unique**.

You have to answer t independent test cases.

Input

The first line of the input contains one integer t ($1 \le t \le 400$) — the number of test cases. Then t test cases follow.

The first line of the test case contains one integer n ($1 \le n \le 50$) — the length of permutation. The second line of the test case contains $2n$ integers $a_1, a_2, \dots, a_{2n}$ ($1 \le a_i \le n$), where a_i is the i-th element of a. It is guaranteed that the array a represents the result of merging of some permutation p with the same permutation p.

Output

For each test case, print the answer: n integers $p_1, p_2, \dots, p_n$ ($1 \le p_i \le n$), representing the initial permutation. It is guaranteed that the answer **exists and is unique**.

input

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

output

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

H. Two Arrays

1 second, 256 megabytes

RedDreamer has an array a consisting of n non-negative integers, and an unlucky integer T.

Let's denote the misfortune of array b having length m as $f(b)$ — the number of pairs of integers (i, j) such that $1 \le i < j \le m$ and $b_i + b_j = T$. *RedDreamer* has to paint each element of a into one of two colors, white and black (for each element, the color is chosen independently), and then create two arrays c and d so that all white elements belong to c, and all black elements belong to d **(it is possible that one of these two arrays becomes empty)**. *RedDreamer* wants to paint the elements in such a way that $f(c) + f(d)$ is **minimum** possible.

For example:

- if $n = 6$, $T = 7$ and $a = [1, 2, 3, 4, 5, 6]$, it is possible to paint the 1-st, the 4-th and the 5-th elements white, and all other elements black. So $c = [1, 4, 5]$, $d = [2, 3, 6]$, and $f(c) + f(d) = 0 + 0 = 0$;
- if $n = 3$, $T = 6$ and $a = [3, 3, 3]$, it is possible to paint the 1-st element white, and all other elements black. So $c = [3]$, $d = [3, 3]$, and $f(c) + f(d) = 0 + 1 = 1$.

Help *RedDreamer* to paint the array optimally!

Input

The first line contains one integer t ($1 \le t \le 1000$) — the number of test cases. Then t test cases follow.

The first line of each test case contains two integers n and T ($1 \le n \le 10^5$, $0 \le T \le 10^9$) — the number of elements in the array and the unlucky integer, respectively.

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

The sum of n over all test cases does not exceed 10^5.

Output

For each test case print n integers: p_1, p_2, ..., p_n (each p_i is either 0 or 1) denoting the colors. If p_i is 0, then a_i is white and belongs to the array c, otherwise it is black and belongs to the array d.

If there are multiple answers that minimize the value of $f(c) + f(d)$, print any of them.

input

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

output

```
1 0 0 1 1 0
1 0 0
```

I. Kuroni and Simple Strings

1 second, 256 megabytes

Now that Kuroni has reached 10 years old, he is a big boy and doesn't like arrays of integers as presents anymore. This year he wants a Bracket sequence as a Birthday present. More specifically, he wants a bracket sequence so complex that no matter how hard he tries, he will not be able to remove a simple subsequence!

We say that a string formed by n characters `'('` or `')'` is **simple** if its length n is even and positive, its first $\frac{n}{2}$ characters are `'('`, and its last $\frac{n}{2}$ characters are `')'`. For example, the strings `()` and `((()))` are simple, while the strings `)(` and `()()` are not simple.

Kuroni will be given a string formed by characters `'('` and `')'` (the given string is not necessarily simple). An operation consists of choosing a subsequence of the characters of the string that forms a simple string and removing all the characters of this subsequence from the string. **Note that this subsequence doesn't have to be continuous**. For example, he can apply the operation to the string `')(()(()))'`, to choose a subsequence of bold characters, as it forms a simple string `'(((())))'`, delete these bold characters from the string and to get `'))()'`.

Kuroni has to perform the minimum possible number of operations on the string, in such a way that no more operations can be performed on the remaining string. The resulting string **does not** have to be empty.

Since the given string is too large, Kuroni is unable to figure out how to minimize the number of operations. Can you help him do it instead?

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

Input

The only line of input contains a string s ($1 \le |s| \le 1000$) formed by characters `'('` and `')'`, where $|s|$ is the length of s.

Output

In the first line, print an integer k — the minimum number of operations you have to apply. Then, print $2k$ lines describing the operations in the following format:

For each operation, print a line containing an integer m — the number of characters in the subsequence you will remove.

Then, print a line containing m integers $1 \le a_1 < a_2 < \cdots < a_m$ — the indices of the characters you will remove. All integers must be less than or equal to the length of the current string, and the corresponding subsequence must form a simple string.

If there are multiple valid sequences of operations with the smallest k, you may print any of them.

input

```
(()((
```

output

```
1
2
1 3
```

input

```
)(
```

output

```
0
```

input

```
(()())
```

output

```
1
4
1 2 5 6
```

In the first sample, the string is '(()(('. The operation described corresponds to deleting the bolded subsequence. The resulting string is '(((', and no more operations can be performed on it. Another valid answer is choosing indices 2 and 3, which results in the same final string.

In the second sample, it is already impossible to perform any operations.

J. Kirill And The Game

2 seconds, 256 megabytes

Kirill plays a new computer game. He came to the potion store where he can buy any potion. Each potion is characterized by two integers — amount of experience and cost. The efficiency of a potion is the ratio of the amount of experience to the cost. Efficiency may be a non-integer number.

For each two integer numbers a and b such that $l \le a \le r$ and $x \le b \le y$ there is a potion with experience a and cost b in the store (that is, there are $(r - l + 1) \cdot (y - x + 1)$ potions).

Kirill wants to buy a potion which has efficiency k. Will he be able to do this?

Input

First string contains five integer numbers l, r, x, y, k ($1 \le l \le r \le 10^7$, $1 \le x \le y \le 10^7$, $1 \le k \le 10^7$).

Output

Print "YES" without quotes if a potion with efficiency exactly k can be bought in the store and "NO" without quotes otherwise.

You can output each of the letters in any register.

input

```
1 10 1 10 1
```

output

```
YES
```

input

```
1 5 6 10 1
```

output

```
NO
```

K. Pair Programming

2 seconds, 512 megabytes

Monocarp and Polycarp are learning new programming techniques. Now they decided to try pair programming.

It's known that they have worked together on the same file for $n + m$ minutes. Every minute exactly one of them made one change to the file. Before they started, there were already k lines written in the file.

Every minute exactly one of them does one of two actions: adds a new line to the end of the file or changes one of its lines.

Monocarp worked in total for n minutes and performed the sequence of actions $[a_1, a_2, \dots, a_n]$. If $a_i = 0$, then he adds a new line to the end of the file. If $a_i > 0$, then he changes the line with the number a_i. Monocarp performed actions strictly in this order: a_1, then a_2, ..., a_n.

Polycarp worked in total for m minutes and performed the sequence of actions $[b_1, b_2, \dots, b_m]$. If $b_j = 0$, then he adds a new line to the end of the file. If $b_j > 0$, then he changes the line with the number b_j. Polycarp performed actions strictly in this order: b_1, then b_2, ..., b_m.

Restore their common sequence of actions of length $n + m$ such that all actions would be correct — there should be no changes to lines that do not yet exist. Keep in mind that in the common sequence Monocarp's actions should form the subsequence $[a_1, a_2, \dots, a_n]$ and Polycarp's — subsequence $[b_1, b_2, \dots, b_m]$. They can replace each other at the computer any number of times.

Let's look at an example. Suppose $k = 3$. Monocarp first changed the line with the number 2 and then added a new line (thus, $n = 2,\ a = [2, 0]$). Polycarp first added a new line and then changed the line with the number 5 (thus, $m = 2,\ b = [0, 5]$).

Since the initial length of the file was 3, in order for Polycarp to change line number 5 two new lines must be added beforehand. Examples of correct sequences of changes, in this case, would be $[0, 2, 0, 5]$ and $[2, 0, 0, 5]$. Changes $[0, 0, 5, 2]$ (wrong order of actions) and $[0, 5, 2, 0]$ (line 5 cannot be edited yet) are not correct.

Input

The first line contains an integer t ($1 \le t \le 1000$). Then t test cases follow. Before each test case, there is an empty line.

Each test case contains three lines. The first line contains three integers k, n, m ($0 \le k \le 100$, $1 \le n, m \le 100$) — the initial number of lines in file and lengths of Monocarp's and Polycarp's sequences of changes respectively.

The second line contains n integers $a_1, a_2, \dots, a_n$ ($0 \le a_i \le 300$).

The third line contains m integers $b_1, b_2, \dots, b_m$ ($0 \le b_j \le 300$).

Output

For each test case print any correct common sequence of Monocarp's and Polycarp's actions of length $n + m$ or -1 if such sequence doesn't exist.

input

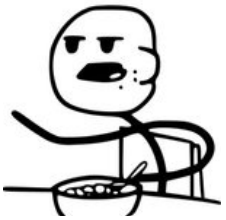
```
5

3 2 2
2 0
0 5

4 3 2
2 0 5
0 6

0 2 2
1 0
2 3

5 4 4
6 0 8 0
0 7 0 9

5 4 1
8 7 8 0
0
```

output

```
2 0 0 5
0 2 0 6 5
-1
0 6 0 7 0 8 0 9
-1
```

L. Where Are My Flakes?

2 seconds, 256 megabytes



One morning the Cereal Guy found out that all his cereal flakes were gone. He found a note instead of them. It turned out that his smart roommate hid the flakes in one of n boxes. The boxes stand in one row, they are numbered from 1 to n from the left to the right. The roommate left hints like "Hidden to the left of the i-th box" ("`To the left of i`"), "Hidden to the right of the i-th box" ("`To the right of i`"). Such hints mean that there are no flakes in the i-th box as well. The Cereal Guy wants to know the minimal number of boxes he necessarily needs to check to find the flakes considering all the hints. Or he wants to find out that the hints are contradictory and the roommate lied to him, that is, no box has the flakes.

Input
The first line contains two integers n and m ($1 \le n \le 1000, 0 \le m \le 1000$) which represent the number of boxes and the number of hints correspondingly. Next m lines contain hints like "`To the left of i`" and "`To the right of i`", where i is integer ($1 \le i \le n$). The hints may coincide.

Output
The answer should contain exactly one integer — the number of boxes that should necessarily be checked or "`-1`" if the hints are contradictory.

input
```
2 1
To the left of 2
```
output
```
1
```

input
```
3 2
To the right of 1
To the right of 2
```
output
```
1
```

input
```
3 1
To the left of 3
```
output
```
2
```

input
```
3 2
To the left of 2
To the right of 1
```
output
```
-1
```

M. Greg and Array

1.5 seconds, 256 megabytes

Greg has an array $a = a_1, a_2, ..., a_n$ and m operations. Each operation looks as: $l_i, r_i, d_i, (1 \le l_i \le r_i \le n)$. To apply operation i to the array means to increase all array elements with numbers $l_i, l_i + 1, ..., r_i$ by value d_i.

Greg wrote down k queries on a piece of paper. Each query has the following form: $x_i, y_i, (1 \le x_i \le y_i \le m)$. That means that one should apply operations with numbers $x_i, x_i + 1, ..., y_i$ to the array.

Now Greg is wondering, what the array a will be after all the queries are executed. Help Greg.

Input
The first line contains integers n, m, k $(1 \le n, m, k \le 10^5)$. The second line contains n integers: $a_1, a_2, ..., a_n$ $(0 \le a_i \le 10^5)$ — the initial array.

Next m lines contain operations, the operation number i is written as three integers: $l_i, r_i, d_i, (1 \le l_i \le r_i \le n), (0 \le d_i \le 10^5)$.

Next k lines contain the queries, the query number i is written as two integers: $x_i, y_i, (1 \le x_i \le y_i \le m)$.

The numbers in the lines are separated by single spaces.

Output
On a single line print n integers $a_1, a_2, ..., a_n$ — the array after executing all the queries. Separate the printed numbers by spaces.

Please, do not use the `%lld` specifier to read or write 64-bit integers in *C++*. It is preferred to use the `cin`, `cout` streams of the `%I64d` specifier.

input
```
3 3 3
1 2 3
1 2 1
1 3 2
2 3 4
1 2
1 3
2 3
```
output
```
9 18 17
```

input
```
1 1 1
1
1 1 1
1 1
```
output
```
2
```

input
```
4 3 6
1 2 3 4
1 2 1
2 3 2
3 4 4
1 2
1 3
2 3
1 2
1 3
2 3
```
output
```
5 18 31 20
```

N. Kuriyama Mirai's Stones

2 seconds, 256 megabytes

Kuriyama Mirai has killed many monsters and got many (namely n) stones. She numbers the stones from 1 to n. The cost of the i-th stone is v_i. Kuriyama Mirai wants to know something about these stones so she will ask you two kinds of questions:

1. She will tell you two numbers, l and r $(1 \le l \le r \le n)$, and you should tell her $\sum_{i=l}^{r} v_i$.
2. Let u_i be the cost of the i-th cheapest stone (the cost that will be on the i-th place if we arrange all the stone costs in non-decreasing order). This time she will tell you two numbers, l and r $(1 \le l \le r \le n)$, and you should tell her $\sum_{i=l}^{r} u_i$.

For every question you should give the correct answer, or Kuriyama Mirai will say "fuyukai desu" and then become unhappy.

Input
The first line contains an integer n $(1 \le n \le 10^5)$. The second line contains n integers: $v_1, v_2, ..., v_n$ $(1 \le v_i \le 10^9)$ — costs of the stones.

The third line contains an integer m ($1 \le m \le 10^5$) — the number of Kuriyama Mirai's questions. Then follow m lines, each line contains three integers $type$, l and r ($1 \le l \le r \le n$; $1 \le type \le 2$), describing a question. If $type$ equal to 1, then you should output the answer for the first question, else you should output the answer for the second one.

Output

Print m lines. Each line must contain an integer — the answer to Kuriyama Mirai's question. Print the answers to the questions in the order of input.

input

```
6
6 4 2 7 2 7
3
2 3 6
1 3 4
1 1 6
```

output

```
24
9
28
```

input

```
4
5 5 2 3
10
1 2 4
2 1 4
1 1 1
2 1 4
2 1 2
1 1 1
1 3 3
1 1 3
1 4 4
1 2 2
```

output

```
10
15
5
15
5
5
2
12
3
5
```

Please note that the answers to the questions may overflow 32-bit integer type.