Codeforces Round 895 (Div. 3)

A. Two Vessels

1 second, 256 megabytes

You have two vessels with water. The first vessel contains a grams of water, and the second vessel contains b grams of water. Both vessels are very large and can hold any amount of water.

You also have an empty cup that can hold **up to** c grams of water.

In one move, you can scoop **up to** c grams of water from any vessel and pour it into **the other** vessel. Note that the mass of water poured in one move **does not have to be an integer**.

What is the minimum number of moves required to make the masses of water in the vessels equal? Note that you cannot perform any actions other than the described moves.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 1000$). The description of the test cases follows.

Each test case consists of a single line containing three integers a, b, and c ($1 \le a, b, c \le 100$) — the mass of water in the vessels and the capacity of the cup, respectively.

Output

For each test case, output a single number — the minimum number of moves required to make the masses of water in the vessels equal. It can be shown, that it is always possible.

input

```
6
3 7 2
17 4 3
17 17 1
17 21 100
1 100 1
97 4 3
```

output

```
1
3
0
1
50
16
```

In the first test case, only one move is enough: if we pour 2 grams of water from the second vessel into the first one, both vessels will contain 5 grams of water.

In the second example test case, three moves are enough:

- Pour 3 grams of water from the first vessel into the second one. After this move, the first vessel will contain $17 - 3 = 14$ grams of water, and the second vessel will contain $4 + 3 = 7$ grams.
- Pour 2 grams of water from the first vessel into the second one. After this move, the first vessel will contain $14 - 2 = 12$ grams of water, and the second vessel will contain $7 + 2 = 9$ grams.
- Finally, pour 1.5 grams of water from the first vessel into the second one. After this move, the first vessel will contain $12 - 1.5 = 10.5$ grams of water, and the second vessel will contain $9 + 1.5 = 10.5$ grams.

Note that this is not the only way to equalize the vessels in 3 moves, but there is no way to do it in 2 moves.

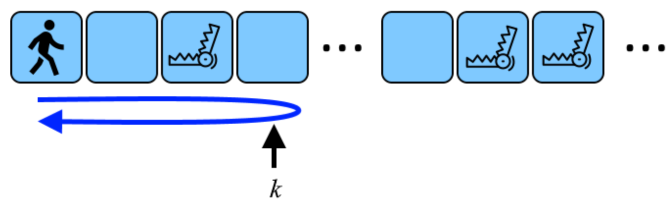
In the third example test case, the vessels initially contain the same amount of water, so no moves are needed. The answer is 0.

B. The Corridor or There and Back Again

2 seconds, 256 megabytes

You are in a corridor that extends infinitely to the right, divided into square rooms. You start in room 1, proceed to room k, and then return to room 1. You can choose the value of k. Moving to an adjacent room takes 1 second.

Additionally, there are n traps in the corridor: the i-th trap is located in room d_i and will be activated s_i seconds **after you enter the room d_i**. Once a trap is activated, you cannot enter or exit a room with that trap.



A schematic representation of a possible corridor and your path to room k and back.

Determine the maximum value of k that allows you to travel from room 1 to room k and then return to room 1 safely.

For instance, if $n = 1$ and $d_1 = 2, s_1 = 2$, you can proceed to room $k = 2$ and return safely (the trap will activate at the moment $1 + s_1 = 1 + 2 = 3$, it can't prevent you to return back). But if you attempt to reach room $k = 3$, the trap will activate at the moment $1 + s_1 = 1 + 2 = 3$, preventing your return (you would attempt to enter room 2 on your way back at second 3, but the activated trap would block you). Any larger value for k is also not feasible. Thus, the answer is $k = 2$.

Input

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

The descriptions of the test cases follow.

The first line of each test case description contains an integer n ($1 \le n \le 100$) — the number of traps.

The following n lines of each test case description present two integers d_i and s_i ($1 \le d_i, s_i \le 200$) — the parameters of a trap (you must leave room d_i strictly before s_i seconds have passed since entering this room). It's possible for multiple traps to occupy a single room (the values of d_i can be repeated).

Output

For each test case, print the maximum value of k that allows you to travel to room k and return to room 1 without encountering an active trap.

input

```
7
1
2 2
3
2 8
4 3
5 2
1
200 200
4
1 20
5 9
3 179
100 1
2
10 1
1 18
2
1 1
1 2
3
1 3
1 1
1 3
```

output

```
2
5
299
9
9
1
1
```

The first test case is explained in the problem statement above.

In the second test case, the second trap prevents you from achieving $k \geq 6$. If $k \geq 6$, the second trap will activate at the moment $3 + s_2 = 3 + 3 = 6$ (the time you enter room 4 plus s_2). In the case of $k \geq 6$, you will return to room 4 at time 7 or later. The trap will be active at that time. It can be shown that room $k = 5$ can be reached without encountering an active trap.

In the third test case, you can make it to room 299 and then immediately return to room 1.

C. Non-coprime Split

1 second, 256 megabytes

You are given two integers $l \leq r$. You need to find **positive** integers a and b such that the following conditions are simultaneously satisfied:

- $l \leq a + b \leq r$
- $\gcd(a, b) \neq 1$

or report that they do not exist.

$\gcd(a, b)$ denotes the greatest common divisor of numbers a and b. For example, $\gcd(6, 9) = 3$, $\gcd(8, 9) = 1$, $\gcd(4, 2) = 2$.

Input

The first line of the input contains an integer t ($1 \leq t \leq 500$) — the number of test cases.

Then the descriptions of the test cases follow.

The only line of the description of each test case contains 2 integers l, r ($1 \leq l \leq r \leq 10^7$).

Output

For each test case, output the integers a, b that satisfy all the conditions on a separate line. If there is no answer, instead output a single number -1.

If there are multiple answers, you can output any of them.

input

```
11
11 15
1 3
18 19
41 43
777 777
8000000 10000000
2000 2023
1791791 1791791
1 4
2 3
9840769 9840769
```

output

```
6 9
-1
14 4
36 6
111 666
4000000 5000000
2009 7
-1
2 2
-1
6274 9834495
```

In the first test case, $11 \leq 6 + 9 \leq 15$, $\gcd(6, 9) = 3$, and all conditions are satisfied. Note that this is not the only possible answer, for example, $\{4, 10\}, \{5, 10\}, \{6, 6\}$ are also valid answers for this test case.

In the second test case, the only pairs $\{a, b\}$ that satisfy the condition $1 \leq a + b \leq 3$ are $\{1, 1\}, \{1, 2\}, \{2, 1\}$, but in each of these pairs $\gcd(a, b)$ equals 1, so there is no answer.

In the third sample test, $\gcd(14, 4) = 2$.

D. Plus Minus Permutation

1 second, 256 megabytes

You are given 3 integers — n, x, y. Let's call the *score* of a permutation[†] $p_1, \ldots, p_n$ the following value:

$$(p_{1 \cdot x} + p_{2 \cdot x} + \ldots + p_{\lfloor \frac{n}{x} \rfloor \cdot x}) - (p_{1 \cdot y} + p_{2 \cdot y} + \ldots + p_{\lfloor \frac{n}{y} \rfloor \cdot y})$$

In other words, the *score* of a permutation is the sum of p_i for all indices i divisible by x, minus the sum of p_i for all indices i divisible by y.

You need to find the maximum possible *score* among all permutations of length n.

For example, if $n = 7$, $x = 2$, $y = 3$, the maximum *score* is achieved by the permutation $[2, \underline{6}, \underline{1}, \underline{7}, 5, \underline{4}, 3]$ and is equal to $(6 + 7 + 4) - (1 + 4) = 17 - 5 = 12$.

[†] A permutation of length n is an array consisting of n distinct integers from 1 to n in any order. For example, $[2, 3, 1, 5, 4]$ is a permutation, but $[1, 2, 2]$ is not a permutation (the number 2 appears twice in the array) and $[1, 3, 4]$ is also not a permutation ($n = 3$, but the array contains 4).

Input

The first line of input contains an integer t ($1 \leq t \leq 10^4$) — the number of test cases.

Then follows the description of each test case.

The only line of each test case description contains 3 integers n, x, y ($1 \leq n \leq 10^9$, $1 \leq x, y \leq n$).

Output

For each test case, output a single integer — the maximum *score* among all permutations of length n.

input

```
8
7 2 3
12 6 3
9 1 9
2 2 2
100 20 50
24 4 6
1000000000 5575 25450
4 4 1
```

output

```
12
-3
44
0
393
87
179179179436104
-6
```

The first test case is explained in the problem statement above.

In the second test case, one of the optimal permutations will be $[12, 11, \underline{2}, 4, 8, \underline{9}, 10, 6, \underline{1}, 5, 3, \underline{7}]$. The *score* of this permutation is $(9 + 7) - (2 + 9 + 1 + 7) = -3$. It can be shown that a *score* greater than -3 can not be achieved. Note that the answer to the problem can be negative.

In the third test case, the *score* of the permutation will be $(p_1 + p_2 + \ldots + p_9) - p_9$. One of the optimal permutations for this case is $[9, 8, 7, 6, 5, 4, 3, 2, 1]$, and its *score* is 44. It can be shown that a *score* greater than 44 can not be achieved.

In the fourth test case, $x = y$, so the *score* of any permutation will be 0.

E. Data Structures Fan

2 seconds, 256 megabytes

You are given an array of integers $a_1, a_2, \ldots, a_n$, as well as a binary string† s consisting of n characters.

Augustin is a big fan of data structures. Therefore, he asked you to implement a data structure that can answer q queries. There are two types of queries:

- "1 l r" ($1 \le l \le r \le n$) — replace each character s_i for $l \le i \le r$ with its opposite. That is, replace all 0 with 1 and all 1 with 0.
- "2 g" ($g \in \{0, 1\}$) — calculate the value of the bitwise XOR of the numbers a_i for all indices i such that $s_i = g$. Note that the XOR of an empty set of numbers is considered to be equal to 0.

Please help Augustin to answer all the queries!

For example, if $n = 4$, $a = [1, 2, 3, 6]$, $s = 1001$, consider the following series of queries:

1. "2 0" — we are interested in the indices i for which $s_i = 0$, since $s = 1001$, these are the indices 2 and 3, so the answer to the query will be $a_2 \oplus a_3 = 2 \oplus 3 = 1$.
2. "1 1 3" — we need to replace the characters s_1, s_2, s_3 with their opposites, so before the query $s = 1001$, and after the query: $s = 0111$.
3. "2 1" — we are interested in the indices i for which $s_i = 1$, since $s = 0111$, these are the indices 2, 3, and 4, so the answer to the query will be $a_2 \oplus a_3 \oplus a_4 = 2 \oplus 3 \oplus 6 = 7$.
4. "1 2 4" — $s = 0111 \to s = 0000$.
5. "2 1" — $s = 0000$, there are no indices with $s_i = 1$, so since the XOR of an empty set of numbers is considered to be equal to 0, the answer to this query is 0.

† A binary string is a string containing only characters 0 or 1.

Input

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

The descriptions of the test cases follow.

The first line of each test case description contains an integer n ($1 \le n \le 10^5$) — the length of the array.

The second line of the test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$).

The third line of the test case contains the binary string s of length n.

The fourth line of the test case contains one integer q ($1 \le q \le 10^5$) — the number of queries.

The subsequent q lines of the test case describe the queries. The first number of each query, $tp \in \{1, 2\}$, characterizes the type of the query: if $tp = 1$, then 2 integers $1 \le l \le r \le n$ follow, meaning that the operation of type 1 should be performed with parameters l, r, and if $tp = 2$, then one integer $g \in \{0, 1\}$ follows, meaning that the operation of type 2 should be performed with parameter g.

It is guaranteed that the sum of n over all test cases does not exceed 10^5, and also that the sum of q over all test cases does not exceed 10^5.

Output

For each test case, and for each query of type 2 in it, output the answer to the corresponding query.

input

```
5
5
1 2 3 4 5
01000
7
2 0
2 1
1 2 4
2 0
2 1
1 1 3
2 1
6
12 12 14 14 5 5
001001
3
2 1
1 2 4
2 1
4
7 7 7 777
1111
3
2 0
1 2 3
2 0
2
1000000000 996179179
11
1
2 1
5
1 42 20 47 7
00011
5
1 3 4
1 1 1
1 3 4
1 2 4
2 0
```

output

```
3 2 6 7 7
11 7
0 0
16430827
47
```

Let's analyze the first test case:

1. "2 0" — we are interested in the indices i for which $s_i = 0$, since $s =$ 01000, these are the indices $1, 3, 4$, and 5, so the answer to the query will be $a_1 \oplus a_3 \oplus a_4 \oplus a_5 = 1 \oplus 3 \oplus 4 \oplus 5 = 3$.
2. "2 1" — we are interested in the indices i for which $s_i = 1$, since $s =$ 01000, the only suitable index is 2, so the answer to the query will be $a_2 = 2$.
3. "1 2 4" — we need to replace the characters s_2, s_3, s_4 with their opposites, so before the query $s =$ 01000, and after the query: $s =$ 00110.
4. "2 0" — we are interested in the indices i for which $s_i = 0$, since $s =$ 00110, these are the indices $1, 2$, and 5, so the answer to the query will be $a_1 \oplus a_2 \oplus a_5 = 1 \oplus 2 \oplus 5 = 6$.
5. "2 1" — we are interested in the indices i for which $s_i = 1$, since $s =$ 00110, these are the indices 3 and 4, so the answer to the query will be $a_3 \oplus a_4 = 3 \oplus 4 = 7$.
6. "1 1 3" — $s =$ 00110 $\to s =$ 11010.
7. "2 1" — we are interested in the indices i for which $s_i = 1$, since $s =$ 11010, these are the indices $1, 2$, and 4, so the answer to the query will be $a_1 \oplus a_2 \oplus a_4 = 1 \oplus 2 \oplus 4 = 7$.

F. Selling a Menagerie

2 seconds, 256 megabytes

You are the owner of a menagerie consisting of n animals numbered from 1 to n. However, maintaining the menagerie is quite expensive, so you have decided to sell it!

It is known that each animal is afraid of exactly one other animal. More precisely, animal i is afraid of animal a_i ($a_i \neq i$). Also, the cost of each animal is known, for animal i it is equal to c_i.

You will sell all your animals in some fixed order. Formally, you will need to choose some permutation[†] $p_1, p_2, \ldots, p_n$, and sell animal p_1 first, then animal p_2, and so on, selling animal p_n last.

When you sell animal i, there are two possible outcomes:

- If animal a_i was sold **before** animal i, you receive c_i money for selling animal i.
- If animal a_i was **not** sold **before** animal i, you receive $2 \cdot c_i$ money for selling animal i. (Surprisingly, animals that are currently afraid are more valuable).

Your task is to choose the order of selling the animals in order to maximize the total profit.

For example, if $a = [3, 4, 4, 1, 3]$, $c = [3, 4, 5, 6, 7]$, and the permutation you choose is $[4, 2, 5, 1, 3]$, then:

- The first animal to be sold is animal 4. Animal $a_4 = 1$ was not sold before, so you receive $2 \cdot c_4 = 12$ money for selling it.
- The second animal to be sold is animal 2. Animal $a_2 = 4$ was sold before, so you receive $c_2 = 4$ money for selling it.
- The third animal to be sold is animal 5. Animal $a_5 = 3$ was not sold before, so you receive $2 \cdot c_5 = 14$ money for selling it.
- The fourth animal to be sold is animal 1. Animal $a_1 = 3$ was not sold before, so you receive $2 \cdot c_1 = 6$ money for selling it.
- The fifth animal to be sold is animal 3. Animal $a_3 = 4$ was sold before, so you receive $c_3 = 5$ money for selling it.

Your total profit, with this choice of permutation, is $12 + 4 + 14 + 6 + 5 = 41$. Note that 41 is **not** the maximum possible profit in this example.

[†] A permutation of length n is an array consisting of n distinct integers from 1 to n in any order. For example, $[2, 3, 1, 5, 4]$ is a permutation, but $[1, 2, 2]$ is not a permutation (2 appears twice in the array) and $[1, 3, 4]$ is also not a permutation ($n = 3$, but 4 is present in the array).

Input

The first line of the input contains an integer t ($1 \le t \le 10^4$) — the number of test cases.

Then follow the descriptions of the test cases.

The first line of each test case description contains an integer n ($2 \le n \le 10^5$) — the number of animals.

The second line of the test case description contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$, $a_i \neq i$) — a_i means the index of the animal that animal i is afraid of.

The third line of the test case description contains n integers $c_1, c_2, \ldots, c_n$ ($1 \le c_i \le 10^9$) — the costs of the animals.

It is guaranteed that the sum of n over all test cases does not exceed 10^5.

Output

Output t lines, each containing the answer to the corresponding test case. The answer should be n integers — the permutation $p_1, p_2, \ldots, p_n$, indicating in which order to sell the animals in order to maximize the profit. If there are multiple possible answers, you can output any of them.

input

```
8
3
2 3 2
6 6 1
8
2 1 4 3 6 5 8 7
1 2 1 2 2 1 2 1
5
2 1 1 1 1
9 8 1 1 1
2
2 1
1000000000 999999999
7
2 3 2 6 4 4 3
1 2 3 4 5 6 7
5
3 4 4 1 3
3 4 5 6 7
3
2 1 1
1 2 2
4
2 1 4 1
1 1 1 1
```

output

```
1 2 3
2 4 5 1 6 3 7 8
3 4 5 1 2
1 2
7 5 1 3 2 6 4
5 3 2 4 1
3 2 1
3 4 1 2
```

G. Replace With Product

1 second, 256 megabytes

Given an array a of n positive integers. You need to perform the following operation **exactly** once:

- Choose 2 integers l and r ($1 \le l \le r \le n$) and replace the subarray $a[l \ldots r]$ with the single element: the product of all elements in the subarray $(a_l \cdot \ldots \cdot a_r)$.

For example, if an operation with parameters $l = 2, r = 4$ is applied to the array $[5, 4, 3, 2, 1]$, the array will turn into $[5, 24, 1]$.

Your task is to maximize the sum of the array after applying this operation. Find the optimal subarray to apply this operation.

Input

Each test consists of multiple test cases. The first line contains a single integer t ($1 \le t \le 10^4$) — the number of test cases. This is followed by the description of the test cases.

The first line of each test case contains a single number n ($1 \le n \le 2 \cdot 10^5$) — the length of the array a.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$).

It is guaranteed that the sum of the values of n for all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output 2 integers l and r ($1 \le l \le r \le n$) — the boundaries of the subarray to be replaced with the product.

If there are multiple solutions, output any of them.

input

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

output

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

In the first test case, after applying the operation with parameters $l = 2, r = 4$, the array $[1, 3, 1, 3]$ turns into $[1, 9]$, with a sum equal to 10. It is easy to see that by replacing any other segment with a product, the sum will be less than 10.

In the second test case, after applying the operation with parameters $l = 3, r = 4$, the array $[1, 1, 2, 3]$ turns into $[1, 1, 6]$, with a sum equal to 8. It is easy to see that by replacing any other segment with a product, the sum will be less than 8.

In the third test case, it will be optimal to choose any operation with $l = r$, then the sum of the array will remain 5, and when applying any other operation, the sum of the array will decrease.