GPC-UPC Simple Segment Tree Contest

A. Xenia and Bit Operations

2 seconds, 256 megabytes

Xenia the beginner programmer has a sequence a, consisting of 2^n non-negative integers: $a_1, a_2, ..., a_{2^n}$. Xenia is currently studying bit operations. To better understand how they work, Xenia decided to calculate some value v for a.

Namely, it takes several iterations to calculate value v. At the first iteration, Xenia writes a new sequence a_1 *or* a_2, a_3 *or* a_4, ..., $a_{2^n - 1}$ *or* a_{2^n}, consisting of $2^{n - 1}$ elements. In other words, she writes down the bit-wise OR of adjacent elements of sequence a. At the second iteration, Xenia writes the bitwise **exclusive** OR of adjacent elements of the sequence obtained after the first iteration. At the third iteration Xenia writes the bitwise OR of the adjacent elements of the sequence obtained after the second iteration. And so on; the operations of bitwise exclusive OR and bitwise OR alternate. In the end, she obtains a sequence consisting of one element, and that element is v.

Let's consider an example. Suppose that sequence $a = (1, 2, 3, 4)$. Then let's write down all the transformations $(1, 2, 3, 4) \rightarrow (1 \text{ or } 2 = 3, 3 \text{ or } 4 = 7) \rightarrow (3 \text{ xor } 7 = 4)$. The result is $v = 4$.

You are given Xenia's initial sequence. But to calculate value v for a given sequence would be too easy, so you are given additional m queries. Each query is a pair of integers p, b. Query p, b means that you need to perform the assignment $a_p = b$. After each query, you need to print the new value v for the new sequence a.

Input

The first line contains two integers n and m $(1 \le n \le 17, 1 \le m \le 10^5)$. The next line contains 2^n integers $a_1, a_2, ..., a_{2^n}$ $(0 \le a_i < 2^{30})$. Each of the next m lines contains queries. The i-th line contains integers p_i, b_i $(1 \le p_i \le 2^n, 0 \le b_i < 2^{30})$ — the i-th query.

Output

Print m integers — the i-th integer denotes value v for sequence a after the i-th query.

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

For more information on the bit operations, you can follow this link: `http://en.wikipedia.org/wiki/Bitwise_operation`

B. Knight Tournament

3 seconds, 256 megabytes

Hooray! Berl II, the king of Berland is making a knight tournament. The king has already sent the message to all knights in the kingdom and they in turn agreed to participate in this grand event.

As for you, you're just a simple peasant. There's no surprise that you slept in this morning and were late for the tournament (it was a weekend, after all). Now you are really curious about the results of the tournament. This time the tournament in Berland went as follows:

- There are n knights participating in the tournament. Each knight was assigned his unique number — an integer from 1 to n.
- The tournament consisted of m fights, in the i-th fight the knights that were still in the game with numbers at least l_i and at most r_i have fought for the right to continue taking part in the tournament.
- After the i-th fight among all participants of the fight only one knight won — the knight number x_i, he continued participating in the tournament. Other knights left the tournament.
- The winner of the last (the m-th) fight (the knight number x_m) became the winner of the tournament.

You fished out all the information about the fights from your friends. Now for each knight you want to know the name of the knight he was conquered by. We think that the knight number b was conquered by the knight number a, if there was a fight with both of these knights present and the winner was the knight number a.

Write the code that calculates for each knight, the name of the knight that beat him.

Input

The first line contains two integers n, m $(2 \le n \le 3 \cdot 10^5; 1 \le m \le 3 \cdot 10^5)$ — the number of knights and the number of fights. Each of the following m lines contains three integers l_i, r_i, x_i $(1 \le l_i < r_i \le n; l_i \le x_i \le r_i)$ — the description of the i-th fight.

It is guaranteed that the input is correct and matches the problem statement. It is guaranteed that at least two knights took part in each battle.

Output

Print n integers. If the i-th knight lost, then the i-th number should equal the number of the knight that beat the knight number i. If the i-th knight is the winner, then the i-th number must equal 0.

input
```
4 3
1 2 1
1 3 3
1 4 4
```
output
```
3 1 4 0
```

input
```
8 4
3 5 4
3 7 6
2 8 8
1 8 1
```
output
```
0 8 4 6 4 8 6 1
```

Consider the first test case. Knights 1 and 2 fought the first fight and knight 1 won. Knights 1 and 3 fought the second fight and knight 3 won. The last fight was between knights 3 and 4, knight 4 won.

C. Enemy is weak

5 seconds, 256 megabytes

The Romans have attacked again. This time they are much more than the Persians but Shapur is ready to defeat them. He says: "A lion is never afraid of a hundred sheep".

Nevertheless Shapur has to find weaknesses in the Roman army to defeat them. So he gives the army a weakness number.

In Shapur's opinion the weakness of an army is equal to the number of triplets i, j, k such that $i < j < k$ and $a_i > a_j > a_k$ where a_x is the power of man standing at position x. The Roman army has one special trait — powers of all the people in it are distinct.

Help Shapur find out how weak the Romans are.

Input

The first line of input contains a single number n $(3 \le n \le 10^6)$ — the number of men in Roman army. Next line contains n different positive integers a_i $(1 \le i \le n, 1 \le a_i \le 10^9)$ — powers of men in the Roman army.

Output

A single integer number, the weakness of the Roman army.

Please, do not use `%lld` specificator to read or write 64-bit integers in C++. It is preffered to use `cout` (also you may use `%I64d`).

input
```
3
3 2 1
```
output
```
1
```

input
```
3
2 3 1
```
output
```
0
```

input
```
4
10 8 3 1
```
output
```
4
```

input
```
4
1 5 4 3
```
output
```
1
```

D. Sereja and Brackets

1 second, 256 megabytes

Sereja has a bracket sequence $s_1, s_2, ..., s_n$, or, in other words, a string s of length n, consisting of characters "(" and ")".

Sereja needs to answer m queries, each of them is described by two integers l_i, r_i $(1 \le l_i \le r_i \le n)$. The answer to the i-th query is the length of the maximum correct bracket subsequence of sequence $s_{l_i}, s_{l_i+1}, ..., s_{r_i}$. Help Sereja answer all queries.

You can find the definitions for a subsequence and a correct bracket sequence in the notes.

Input

The first line contains a sequence of characters $s_1, s_2, ..., s_n$ $(1 \le n \le 10^6)$ without any spaces. Each character is either a "(" or a ")". The second line contains integer m $(1 \le m \le 10^5)$ — the number of queries. Each of the next m lines contains a pair of integers. The i-th line contains integers l_i, r_i $(1 \le l_i \le r_i \le n)$ — the description of the i-th query.

Output

Print the answer to each question on a single line. Print the answers in the order they go in the input.

input
```
())(())(())(
7
1 1
2 3
1 2
1 12
8 12
5 11
2 10
```
output
```
0
0
2
10
4
6
6
```

A *subsequence* of length $|x|$ of string $s = s_1s_2... s_{|s|}$ (where $|s|$ is the length of string s) is string $x = s_{k_1}s_{k_2}... s_{k_{|x|}}$ $(1 \le k_1 < k_2 < ... < k_{|x|} \le |s|)$.

A *correct bracket sequence* is a bracket sequence that can be transformed into a correct aryphmetic expression by inserting characters "1" and "+" between the characters of the string. For example, bracket sequences "()()", "(())" are correct (the resulting expressions "(1)+(1)", "((1+1)+1)"), and ")(" and "(" are not.

For the third query required sequence will be «()».

For the fourth query required sequence will be «()(())(())».

E. Ant colony

1 second, 256 megabytes

Mole is hungry again. He found one ant colony, consisting of n ants, ordered in a row. Each ant i $(1 \le i \le n)$ has a strength s_i.

In order to make his dinner more interesting, Mole organizes a version of «Hunger Games» for the ants. He chooses two numbers l and r $(1 \le l \le r \le n)$ and each pair of ants with indices between l and r (inclusively) will fight. When two ants i and j fight, ant i gets one battle point only if s_i divides s_j (also, ant j gets one battle point only if s_j divides s_i).

After all fights have been finished, Mole makes the ranking. An ant i, with v_i battle points obtained, is going to be freed only if $v_i = r - l$, or in other words only if it took a point in every fight it participated. After that, Mole eats the rest of the ants. Note that there can be many ants freed or even none.

In order to choose the best sequence, Mole gives you t segments $[l_i, r_i]$ and asks for each of them how many ants is he going to eat if those ants fight.

Input

The first line contains one integer n $(1 \le n \le 10^5)$, the size of the ant colony.

The second line contains n integers $s_1, s_2, ..., s_n$ ($1 \le s_i \le 10^9$), the strengths of the ants.

The third line contains one integer t ($1 \le t \le 10^5$), the number of test cases.

Each of the next t lines contains two integers l_i and r_i ($1 \le l_i \le r_i \le n$), describing one query.

Output

Print to the standard output t lines. The i-th line contains number of ants that Mole eats from the segment $[l_i, r_i]$.

input
```
5
1 3 2 4 2
4
1 5
2 5
3 5
4 5
```
output
```
4
4
1
1
```

In the first test battle points for each ant are $v = [4, 0, 2, 0, 2]$, so ant number 1 is freed. Mole eats the ants $2, 3, 4, 5$.

In the second test case battle points are $v = [0, 2, 0, 2]$, so no ant is freed and all of them are eaten by Mole.

In the third test case battle points are $v = [2, 0, 2]$, so ants number 3 and 5 are freed. Mole eats **only the ant 4**.

In the fourth test case battle points are $v = [0, 1]$, so ant number 5 is freed. Mole eats the ant 4.

F. Misha and Permutations Summation

2 seconds, 256 megabytes

Let's define the sum of two permutations p and q of numbers $0, 1, ..., (n - 1)$ as permutation $Perm((Ord(p) + Ord(q)) \bmod n!)$, where $Perm(x)$ is the x-th lexicographically permutation of numbers $0, 1, ..., (n - 1)$ (counting from zero), and $Ord(p)$ is the number of permutation p in the lexicographical order.

For example, $Perm(0) = (0, 1, ..., n - 2, n - 1)$, $Perm(n! - 1) = (n - 1, n - 2, ..., 1, 0)$

Misha has two permutations, p and q. Your task is to find their sum.

Permutation $a = (a_0, a_1, ..., a_{n - 1})$ is called to be lexicographically smaller than permutation $b = (b_0, b_1, ..., b_{n - 1})$, if for some k following conditions hold: $a_0 = b_0, a_1 = b_1, ..., a_{k - 1} = b_{k - 1}, a_k < b_k$.

Input

The first line contains an integer n ($1 \le n \le 200\,000$).

The second line contains n distinct integers from 0 to $n - 1$, separated by a space, forming permutation p.

The third line contains n distinct integers from 0 to $n - 1$, separated by spaces, forming permutation q.

Output

Print n distinct integers from 0 to $n - 1$, forming the sum of the given permutations. Separate the numbers by spaces.

input
```
2
0 1
0 1
```
output
```
0 1
```

input
```
2
0 1
1 0
```
output
```
1 0
```

input
```
3
1 2 0
2 1 0
```
output
```
1 0 2
```

Permutations of numbers from 0 to 1 in the lexicographical order: $(0, 1), (1, 0)$.

In the first sample $Ord(p) = 0$ and $Ord(q) = 0$, so the answer is $Perm((0 + 0) \bmod 2) = Perm(0) = (0, 1)$.

In the second sample $Ord(p) = 0$ and $Ord(q) = 1$, so the answer is $Perm((0 + 1) \bmod 2) = Perm(1) = (1, 0)$.

Permutations of numbers from 0 to 2 in the lexicographical order: $(0, 1, 2), (0, 2, 1), (1, 0, 2), (1, 2, 0), (2, 0, 1), (2, 1, 0)$.

In the third sample $Ord(p) = 3$ and $Ord(q) = 5$, so the answer is $Perm((3 + 5) \bmod 6) = Perm(2) = (1, 0, 2)$.

G. Pathwalks

1 second, 256 megabytes

You are given a directed graph with n nodes and m edges, with all edges having a certain weight.

There might be multiple edges and self loops, and the graph can also be disconnected.

You need to choose a path (possibly passing through same vertices multiple times) in the graph such that the weights of the edges are in strictly increasing order, and these edges come in the order of input. Among all such paths, you need to find the the path that has the maximum possible number of edges, and report this value.

Please note that the edges picked don't have to be consecutive in the input.

Input

The first line contains two integers n and m ($1 \le n \le 100000, 1 \le m \le 100000$) — the number of vertices and edges in the graph, respectively.

m lines follows.

The i-th of these lines contains three space separated integers a_i, b_i and w_i ($1 \le a_i, b_i \le n$, $0 \le w_i \le 100000$), denoting an edge from vertex a_i to vertex b_i having weight w_i

Output

Print one integer in a single line — the maximum number of edges in the path.

input
```
3 3
3 1 3
1 2 1
2 3 2
```
output
```
2
```

input
```
5 5
1 3 2
3 2 3
3 4 5
5 4 0
4 5 8
```
output
```
3
```

The answer for the first sample input is 2: $1 \to 2 \to 3$. Note that you cannot traverse $1 \to 2 \to 3 \to 1$ because edge $3 \to 1$ appears earlier in the input than the other two edges and hence cannot be picked/traversed after either of the other two edges.

In the second sample, it's optimal to pick 1-st, 3-rd and 5-th edges to get the optimal answer: $1 \to 3 \to 4 \to 5$.

H. XOR on Segment

4 seconds, 256 megabytes

You've got an array a, consisting of n integers $a_1, a_2, ..., a_n$. You are allowed to perform two operations on this array:

1. Calculate the sum of current array elements on the segment $[l, r]$, that is, count value $a_l + a_{l+1} + ... + a_r$.
2. Apply the xor operation with a given number x to each array element on the segment $[l, r]$, that is, execute $a_l = a_l \oplus x, a_{l+1} = a_{l+1} \oplus x, \dots, a_r = a_r \oplus x$. This operation changes exactly $r - l + 1$ array elements.

Expression $x \oplus y$ means applying bitwise xor operation to numbers x and y. The given operation exists in all modern programming languages, for example in language *C++* and *Java* it is marked as "^", in *Pascal* — as "xor".

You've got a list of m operations of the indicated type. Your task is to perform all given operations, for each sum query you should print the result you get.

Input

The first line contains integer n $(1 \le n \le 10^5)$ — the size of the array. The second line contains space-separated integers $a_1, a_2, ..., a_n$ $(0 \le a_i \le 10^6)$ — the original array.

The third line contains integer m $(1 \le m \le 5 \cdot 10^4)$ — the number of operations with the array. The i-th of the following m lines first contains an integer t_i $(1 \le t_i \le 2)$ — the type of the i-th query. If $t_i = 1$, then this is the query of the sum, if $t_i = 2$, then this is the query to change array elements. If the i-th operation is of type 1, then next follow two integers l_i, r_i $(1 \le l_i \le r_i \le n)$. If the i-th operation is of type 2, then next follow three integers l_i, r_i, x_i $(1 \le l_i \le r_i \le n, 1 \le x_i \le 10^6)$. The numbers on the lines are separated by single spaces.

Output

For each query of type 1 print in a single line the sum of numbers on the given segment. Print the answers to the queries in the order in which the queries go in the input.

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, or the `%I64d` specifier.

input
```
5
4 10 3 13 7
8
1 2 4
2 1 3 3
1 2 4
1 3 3
2 2 5 5
1 1 5
2 1 2 10
1 2 3
```
output
```
26
22
0
34
11
```

input
```
6
4 7 4 0 7 3
5
2 2 3 8
1 1 5
2 3 5 1
2 4 5 6
1 2 3
```
output
```
38
28
```

I. Infinite Inversions

2 seconds, 256 megabytes

There is an infinite sequence consisting of all positive integers in the increasing order: $p = \{1, 2, 3, ...\}$. We performed n *swap* operations with this sequence. A $swap(a, b)$ is an operation of swapping the elements of the sequence on positions a and b. Your task is to find the number of inversions in the resulting sequence, i.e. the number of such index pairs (i, j), that $i < j$ and $p_i > p_j$.

Input

The first line contains a single integer n $(1 \le n \le 10^5)$ — the number of *swap* operations applied to the sequence.

Each of the next n lines contains two integers a_i and b_i $(1 \le a_i, b_i \le 10^9, a_i \ne b_i)$ — the arguments of the *swap* operation.

Output

Print a single integer — the number of inversions in the resulting sequence.

input
```
2
4 2
1 4
```
output
```
4
```

input

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

output

```
15
```

In the first sample the sequence is being modified as follows: $\{1, 2, 3, 4, 5, \ldots\} \to \{1, 4, 3, 2, 5, \ldots\} \to \{2, 4, 3, 1, 5 \ldots\}$. It has 4 inversions formed by index pairs $(1, 4)$, $(2, 3)$, $(2, 4)$ and $(3, 4)$.

J. Vika and Segments

2 seconds, 256 megabytes

Vika has an infinite sheet of squared paper. Initially all squares are white. She introduced a two-dimensional coordinate system on this sheet and drew n black horizontal and vertical segments parallel to the coordinate axes. All segments have width equal to 1 square, that means every segment occupy some set of neighbouring squares situated in one row or one column.

Your task is to calculate the number of painted cells. If a cell was painted more than once, it should be calculated exactly once.

Input

The first line of the input contains a single integer n $(1 \le n \le 100\,000)$ — the number of segments drawn by Vika.

Each of the next n lines contains four integers x_1, y_1, x_2 and y_2 $(-10^9 \le x_1, y_1, x_2, y_2 \le 10^9)$ — the coordinates of the endpoints of the segments drawn by Vika. It is guaranteed that all the segments are parallel to coordinate axes. Segments may touch, overlap and even completely coincide.

Output

Print the number of cells painted by Vika. If a cell was painted more than once, it should be calculated exactly once in the answer.

input

```
3
0 1 2 1
1 4 1 2
0 3 2 3
```

output

```
8
```

input

```
4
-2 -1 2 -1
2 1 -2 1
-1 -2 -1 2
1 2 1 -2
```

output

```
16
```

In the first sample Vika will paint squares $(0, 1)$, $(1, 1)$, $(2, 1)$, $(1, 2)$, $(1, 3)$, $(1, 4)$, $(0, 3)$ and $(2, 3)$.

K. Yaroslav and Divisors

2 seconds, 256 megabytes

Yaroslav has an array $p = p_1, p_2, \ldots, p_n$ $(1 \le p_i \le n)$, consisting of n distinct integers. Also, he has m queries:

- Query number i is represented as a pair of integers l_i, r_i $(1 \le l_i \le r_i \le n)$.
- The answer to the query l_i, r_i is the number of pairs of integers q, w $(l_i \le q, w \le r_i)$ such that p_q is the divisor of p_w.

Help Yaroslav, answer all his queries.

Input

The first line contains the integers n and m $(1 \le n, m \le 2 \cdot 10^5)$. The second line contains n distinct integers $p_1, p_2, \ldots, p_n$ $(1 \le p_i \le n)$. The following m lines contain Yaroslav's queries. The i-th line contains integers l_i, r_i $(1 \le l_i \le r_i \le n)$.

Output

Print m integers — the answers to Yaroslav's queries in the order they appear in the input.

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 or the `%I64d` specifier.

input

```
1 1
1
1 1
```

output

```
1
```

input

```
10 9
1 2 3 4 5 6 7 8 9 10
1 10
2 9
3 8
4 7
5 6
2 2
9 10
5 10
4 10
```

output

```
27
14
8
4
2
1
2
7
9
```

L. Domino Principle

2 seconds, 256 megabytes

Vasya is interested in arranging dominoes. He is fed up with common dominoes and he uses the dominoes of different heights. He put n dominoes on the table along one axis, going from left to right. Every domino stands perpendicular to that axis so that the axis passes through the center of its base. The i-th domino has the coordinate x_i and the height h_i. Now Vasya wants to learn for every domino, how many dominoes will fall if he pushes it to the right. Help him do that.

Consider that a domino falls if it is touched strictly above the base. In other words, the fall of the domino with the initial coordinate x and height h leads to the fall of all dominoes on the segment $[x + 1, x + h - 1]$.

Input
The first line contains integer n ($1 \le n \le 10^5$) which is the number of dominoes. Then follow n lines containing two integers x_i and h_i ($-10^8 \le x_i \le 10^8, 2 \le h_i \le 10^8$) each, which are the coordinate and height of every domino. No two dominoes stand on one point.

Output
Print n space-separated numbers z_i — the number of dominoes that will fall if Vasya pushes the i-th domino to the right (including the domino itself).

input
```
4
16 5
20 5
10 10
18 2
```
output
```
3 1 4 1
```

input
```
4
0 10
1 5
9 10
15 10
```
output
```
4 1 2 1
```

M. Queries

2.25 seconds, 128 megabytes

XORin discovers an interesting function called Elf. XORina has given XORin an array A of N integers and Q queries. The queries are of 2 types:

1 p x - the element on the position p changes his value to x (A_p=x)

2 a b - find the sum of Elf(i,j) for each $a \le i \le j \le b$, where Elf(i,j)=A_i xor A_{i+1} xor ... xor A_j.

Your task is to print the responses to the 2nd type of queries.

Input
The first line of the input contains n, the size of the array. The second line of the input contains m, the number of queries ($1 \le n, m \le 100000$). The third line of the input contains n numbers, the initial elements of the array (each of them is between 1 and 1000). On the next m lines each line contains a query. The first type of query is 1 p x ($1 \le p \le n$ and $0 \le x \le 1000$). The second type of query is 2 a b ($1 \le a \le b \le n$).

Output
The output contains the answer to the second type of queries, each answer on a line. Print the answer modulo 4001.

input
```
4
8
1 2 3 4
2 1 2
1 1 2
2 1 3
2 1 4
1 3 7
2 1 3
1 4 5
2 1 4
```
output
```
6
11
34
23
32
```

N. SUM and REPLACE

2 seconds, 256 megabytes

Let $D(x)$ be the number of positive divisors of a positive integer x. For example, $D(2) = 2$ (2 is divisible by 1 and 2), $D(6) = 4$ (6 is divisible by 1, 2, 3 and 6).

You are given an array a of n integers. You have to process two types of queries:

1. REPLACE l r — for every $i \in [l, r]$ replace a_i with $D(a_i)$;
2. SUM l r — calculate $\sum_{i=l}^{r} a_i$.

Print the answer for each SUM query.

Input
The first line contains two integers n and m ($1 \le n, m \le 3 \cdot 10^5$) — the number of elements in the array and the number of queries to process, respectively.

The second line contains n integers $a_1, a_2, ..., a_n$ ($1 \le a_i \le 10^6$) — the elements of the array.

Then m lines follow, each containing 3 integers t_i, l_i, r_i denoting i-th query. If $t_i = 1$, then i-th query is REPLACE l_i r_i, otherwise it's SUM l_i r_i ($1 \le t_i \le 2$, $1 \le l_i \le r_i \le n$).

There is at least one SUM query.

Output
For each SUM query print the answer to it.

input
```
7 6
6 4 1 10 3 2 4
2 1 7
2 4 5
1 3 5
2 4 4
1 5 7
2 1 7
```
output
```
30
13
4
22
```

O. Pathwalks Online

1 second, 256 megabytes

This is the online version of Pathwalks Codeforces problem.

You are given a directed graph with n nodes and m edges, with all edges having a certain weight.

There might be multiple edges and self loops, and the graph can also be disconnected.

You need to choose a path (possibly passing through same vertices multiple times) in the graph such that the weights of the edges are in strictly increasing order, and these edges come in the order of input. Among all such paths, you need to find the the path that has the maximum possible number of edges, and report this value **every time a new edge is added**.

Please note that the edges picked don't have to be consecutive in the input.

Input

The first line of input contains two integers n and m $(1 \le n \le 10^5, 1 \le m \le 10^5)$ — the number of vertices and edges in the graph, respectively.

Then, m lines follow.

The i-th line contains three integers a'_i, b'_i and w'_i $(0 \le a'_i, b'_i, w'_i \le 2 \cdot 10^5)$ — The encrypted values of the source node, target node and the edge weight.

Suppose that the answer for the last edge added is X (initially $X = 0$).

You can compute the original values of a_i, b_i and w_i with:

$$a_i = a'_i \oplus X$$

$$b_i = b'_i \oplus X$$

$$w_i = w'_i \oplus X$$

Where $\oplus$ is the Bitwise XOR operation. It holds that $1 \le a_i, b_i \le n$ and $0 \le w_i \le 10^5$.

Output

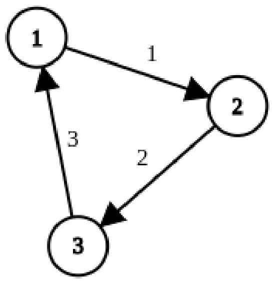
Print m lines — The i-th line must contain the maximum number of edges of a strictly increasing weight path using the first i edges of the input.

input
```
3 3
3 1 3
0 3 0
3 2 3
```
output
```
1
1
2
```

input
```
5 5
1 3 2
2 3 2
1 6 7
7 6 2
6 7 10
```
output
```
1
2
2
2
3
```

For the first sample case, the decrypted edges are:

```
3 3
3 1 3
1 2 1
2 3 2
```
And the graph looks like this:



P. Mass Change Queries

3 seconds, 512 megabytes

You are given an array a consisting of n integers. You have to process q queries to this array; each query is given as four numbers l, r, x and y, denoting that for every i such that $l \le i \le r$ and $a_i = x$ you have to set a_i equal to y.

Print the array after all queries are processed.

Input

The first line contains one integer n $(1 \le n \le 200000)$ — the size of array a.

The second line contains n integers $a_1, a_2, \ldots, a_n$ $(1 \le a_i \le 100)$ — the elements of array a.

The third line contains one integer q $(1 \le q \le 200000)$ — the number of queries you have to process.

Then q lines follow. i-th line contains four integers l, r, x and y denoting i-th query $(1 \le l \le r \le n, 1 \le x, y \le 100)$.

Output

Print n integers — elements of array a after all changes are made.

input
```
5
1 2 3 4 5
3
3 5 3 5
1 5 5 1
1 5 1 5
```
output
```
5 2 5 4 5
```

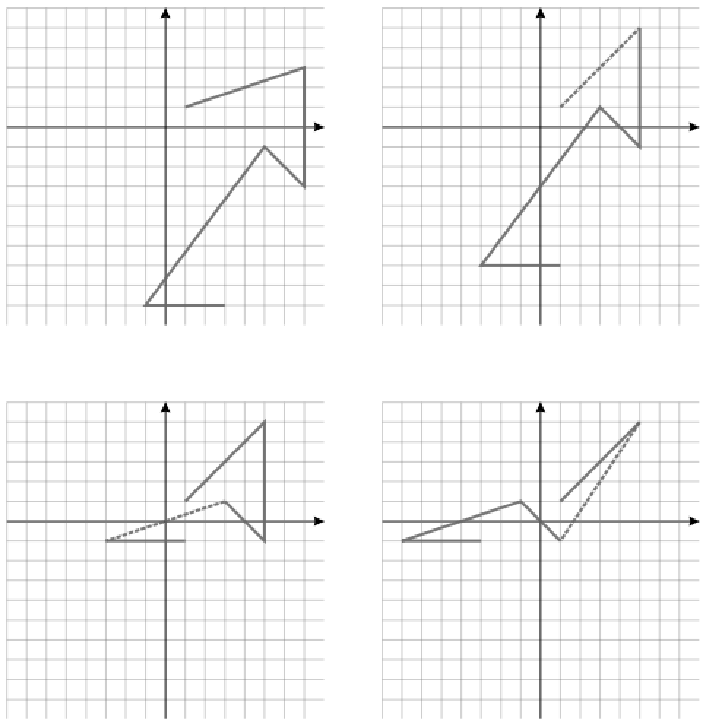
Q. Ruka

2 seconds, 512 megabytes

Stjepan is programming a robot arm that can use a chalk to draw on a blackboard located in a standard coordinate plane (x coordinate increases to the right, y coordinate increases upwards).

The robot arm's plan is an array of precisely n vectors $(x_1, x_2), \dots, (x_n, y_n)$ where each x_i and y_i are even integers. The plan is executed by the robot arm starting from point $(1, 1)$ and making n steps: in the i-th step, the arm moves the chalk from the current point (x, y) straight to the point $(x + x_i, y + y_i)$. Therefore, the robot arm is drawing a kind of a broken line in the coordinate plane, and the segments of that broken line are the given vectors.

While Stjepan is devising and changing his plan, sometimes he wants to know how many times the chalk will **go over the coordinate axes**. Write a programme that will simulate the process of changing the plan and that will give answers to Stjepan's queries.


The layout of the plan on each 'Q' command in the second test case. The dotted line marks the segment that was most recently changed.

Let us assume that Stjepan wrote down his plan in a text file that consists of n lines – the i-th line contains the vector (x_i, y_i). Initially, Stjepan's cursor is located at the first line of the file. Your programme should simulate the following commands:

- 'B' — the cursor moves to the previous line (if it's already located at the first line, nothing happens).
- 'F' — the cursor moves to the next line (if it's already located at the last line, nothing happens).
- 'C nx ny' — where nx and ny are even integers. The row of the file where the cursor is located at changes in a way that the current vector is replaced with the vector (nx, ny).
- 'Q' — you need to output how many times the dotted line which is described by the current plan went over the coordinate axes. If the dotted line goes through the starting point, then we count that as two times going over the coordinate axes.

Input

The first line of input contains the integer n $(1 \leq n \leq 10^5)$ — the number of vectors in the plan. The i-th of the following n lines contains two even integers x_i and y_i $(-500 \leq x_i, y_i \leq 500)$ separated by a single space — the coordinates of the i-th vector in the initial plan.

The following line contains the integer m $(1 \leq m \leq 3 \cdot 10^5)$ — the number of commands which execution you need to simulate.

Each of the following m lines contains a single command. A command is either one of the uppercase letters 'B', 'F' or 'Q' or an expression in the form 'C nx ny' where nx and ny are even integers described in the task $(-500 \leq nx, ny, \leq 500)$.

Output

For each 'Q' command from the input, you must output its result in a single line. The results need to be printed in the order which the commands appear in the input.

input

```
6
2 2
2 -6
-2 -4
-6 4
10 -10
-8 12
16
Q
C -4 -4
F
F
Q
F
C 6 -2
B
B
B
Q
C 0 6
Q
F
C -8 4
Q
```

output

```
4
4
3
1
5
```

input

```
5
6 2
0 -6
-2 2
-6 -8
4 0
12
Q
C 4 4
Q
F
F
F
C -6 -2
Q
B
B
C -4 -6
Q
```

output

```
3
5
5
4
```

R. Order statistic set

0.5 seconds, 512 megabytes

In this problem, you have to maintain a dynamic set of numbers which support the two fundamental operations:

- $INSERT(S, x)$: if x is not in S, insert x into S.
- $DELETE(S, x)$: if x is in S, delete x from S.

and the two type of query operations:

- $K - TH(S, k)$: return the k-th smallest element of S
- $COUNT(S, x)$: return the number of elements of S **smaller than** x.

Initially, your dynamic set is empty and you will have to make q operations. Print the answer for each query operation.

Input

The first line of input contains an integer q $(1 \leq q \leq 2 \cdot 10^5)$ — The number of operations.

The next q lines contain a character c and an integer x $(c \in \{'I', 'D', 'K', 'C'\})$ — The initial letter of the operation type and the parameter for the operation. If $c =' K'$ it will hold that $1 \leq x \leq 10^9$, in any other case it will hold that $|x| \leq 10^9$.

Output

Print the answer to each query operation in consecutive lines. In the case of the $K - TH$ operation, if there is no k-th smallest element, print "invalid".

input

```
8
I -1
I -1
I 2
C 0
K 2
D -1
K 1
K 2
```

output

```
1
2
2
invalid
```

S. Running on Fumes - Chapter 1

7 seconds, 512 megabytes

This problem is from Qualification Round of Facebook Hacker Cup 2020.

Note: This problem shares similarities with Chapter 2. The solution to either chapter may help with solving the other, so please consider reading both first.

You just landed yourself a gig as a delivery driver for a nationwide supply chain. You've been assigned a series of long-haul jobs, so it's time to get to work.

For each job, you will be provided with a map of the relevant region, which includes N cities (numbered from 1 to N) and $N - 1$ two-way roads running amongst them. The cities are connected by roads in a single line, such that there's a road between each pair of consecutive cities. In other words, cities i and j are directly connected by a road if and only if $|i - j| = 1$.

You will begin in city 1 with a shipment of supplies to be delivered to city N, and with a full gas tank having a capacity of M gallons. You will then have two options at each point in time:

1. Drive along a road from your current city to an adjacent one, using up 1 gallon of gas. You may not do this if your tank is empty, but it's fine if your tank becomes empty as a result.
2. Fill your tank all the way back up to M gallons of gas at a cost of C_i dollars, where i is your current city. Note that the cost is independent of how much gas your tank had before refueling. You may not do this if city i has no gas station (indicated with $C_i = 0$).

Determine the minimum cost required to arrive at city N, if it's possible at all.

Input

Input begins with an integer T $(1 \leq T \leq 120)$, the number of long-haul jobs you've been assigned. For each job there is first a line containing the space-separated integers N and M $(2 \leq N \leq 10^6, 1 \leq M \leq N)$. Then, N lines follow, the i-th of which contains the single integer C_i $(0 \leq C_i \leq 10^9)$.

The sum of N across all jobs is at most $4 \cdot 10^6$.

Output

For the i-th job, output a line containing "`Case #i: `" followed by a single integer, the minimum cost in dollars to get from city 1 to city N, or -1 if it's impossible.

input

```
7
5 3
0
20
30
0
10
5 2
0
20
30
0
10
5 1
0
20
30
0
10
4 1
99
88
77
66
4 4
99
88
77
66
6 2
0
0
20
30
0
10
12 3
0
1
4
7
0
5
9
8
0
3
0
6
```

output

```
Case #1: 20
Case #2: 30
Case #3: -1
Case #4: 165
Case #5: 0
Case #6: 50
Case #7: 19
```

T. Running on Fumes - Chapter 2

7 seconds, 512 megabytes

This problem is from Qualification Round of Facebook Hacker Cup 2020.

Note: This problem shares similarities with Chapter 1. The solution to either chapter may help with solving the other, so please consider reading both first.

After working for many months as a delivery driver for a nationwide supply chain, you've finally received a promotion! This promotion has come with a nice pay bump, but also with the responsibility of serving regions with more complex maps. With no opportunity for a vacation, you've promptly been assigned a new series of long-haul jobs.

For each job, you will be provided with a map of the relevant region, which includes N cities (numbered from 1 to N) and $N-1$ two-way roads running amongst them. In computer science terms, the map is organized like a tree rooted at city 1, such each city i is connected to its parent city P_i by a road (aside from city 1, which has no parent, with P_1 given as 0). It's possible to drive from any city to any other city by following a sequence of roads. Note that each road may be driven along in either direction (either from city i to city P_i, or vice versa).

You will begin in a given city A with a shipment of supplies to be delivered to another given city B, and with a full gas tank having a capacity of M gallons. You will then have two options at each point in time:

1. Drive along a road from your current city to an adjacent one, using up 1 gallon of gas. You may not do this if your tank is empty, but it's fine if your tank becomes empty as a result.
2. Fill your tank all the way back up to M gallons of gas at a cost of C_i dollars, where i is your current city. Note that the cost is independent of how much gas your tank had before refueling. You may not do this if city i has no gas station (indicated with $C_i = 0$).

Determine the minimum cost required to complete your delivery, if it's possible at all.

Input

Input begins with an integer T $(1 \le T \le 120)$, the number of long-haul jobs you've been assigned. For each job there is first a line containing the space-separated integers N and M $(2 \le N \le 10^6, 1 \le M \le N)$. Then, N lines follow, the i-th of which contains the integers P_i and C_i $(0 \le P_i \le N, 0 \le C_i \le 10^9)$.

The sum of N across all jobs is at most $3 \cdot 10^6$.

Output

For the i-th job, output a line containing "`Case #i: `" followed by a single integer, the minimum cost in dollars to get from city 1 to city N, or -1 if it's impossible.

input

```
6
4 1 2 3
0 10
4 20
4 30
1 40
8 1 5 3
0 30
3 20
1 0
7 10
8 0
1 0
6 20
6 30
8 2 5 3
0 30
3 20
1 0
7 10
8 0
1 0
6 20
6 30
8 3 5 3
0 30
3 20
1 0
7 10
8 0
1 0
6 20
6 30
15 1 3 14
0 28
1 18
1 17
2 8
4 13
4 2
2 6
5 37
8 37
9 21
6 22
2 33
1 11
8 5
12 8
15 5 11 12
0 0
7 28
1 19
6 5
15 0
5 0
15 0
1 12
8 6
5 20
2 0
4 0
6 10
4 8
1 22
```

output

```
Case #1: 40
Case #2: -1
Case #3: 60
Case #4: 20
Case #5: 104
Case #6: 17
```