Deltix Round, Autumn 2021 (open for everyone, rated, Div. 1 + Div. 2)

A. Divide and Multiply

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

William has array of n numbers $a_1, a_2, \ldots, a_n$. He can perform the following sequence of operations **any number of times**:

1. Pick any two items from array a_i and a_j, where a_i must be a multiple of 2
2. $a_i = \frac{a_i}{2}$
3. $a_j = a_j \cdot 2$

Help William find the maximal sum of array elements, which he can get by performing the sequence of operations described above.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). Description of the test cases follows.

The first line of each test case contains an integer n $(1 \le n \le 15)$, the number of elements in William's array.

The second line contains n integers $a_1, a_2, \ldots, a_n$ $(1 \le a_i < 16)$, the contents of William's array.

Output

For each test case output the maximal sum of array elements after performing an optimal sequence of operations.

Example

input
5 3 6 4 2 5 1 2 3 4 5 1 10 3 2 3 4 15 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
output
50 46 10 26 35184372088846

Note

In the first example test case the optimal sequence would be:

1. Pick $i = 2$ and $j = 1$. After performing a sequence of operations $a_2 = \frac{4}{2} = 2$ and $a_1 = 6 \cdot 2 = 12$, making the array look as: [12, 2, 2].
2. Pick $i = 2$ and $j = 1$. After performing a sequence of operations $a_2 = \frac{2}{2} = 1$ and $a_1 = 12 \cdot 2 = 24$, making the array look as: [24, 1, 2].

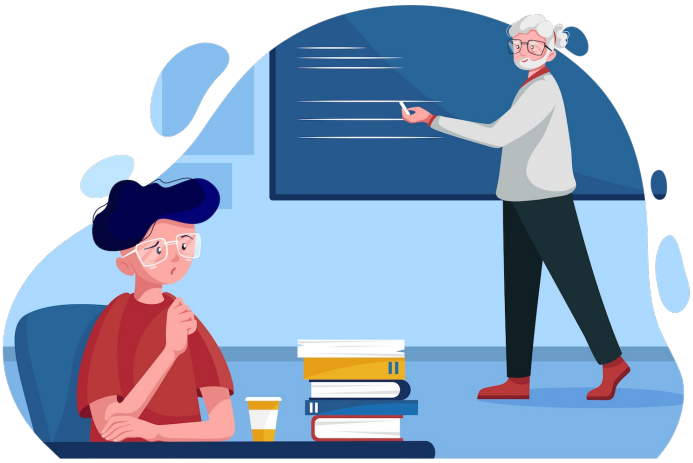
3. Pick $i = 3$ and $j = 1$. After performing a sequence of operations $a_3 = \frac{2}{2} = 1$ and $a_1 = 24 \cdot 2 = 48$, making the array look as: [48, 1, 1].

The final answer $48 + 1 + 1 = 50$.

In the third example test case there is no way to change the sum of elements, so the answer is 10.

B. William the Vigilant

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



Before becoming a successful trader William got a university degree. During his education an interesting situation happened, after which William started to listen to homework assignments much more attentively. What follows is the correct formal description of the homework assignment:

You are given a string s of length n only consisting of characters "a", "b" and "c". There are q queries of format (pos, c), meaning replacing the element of string s at position pos with character c. After each query you must output the minimal number of characters in the string, which have to be replaced, so that the string doesn't contain string "abc" as a **substring**. A valid replacement of a character is replacing it with "a", "b" or "c".

A string x is a substring of a string y if x can be obtained from y by deletion of several (possibly, zero or all) characters from the beginning and several (possibly, zero or all) characters from the end.

Input

The first line contains two integers n and q $(1 \leq n, q \leq 10^5)$, the length of the string and the number of queries, respectively.

The second line contains the string s, consisting of characters "a", "b" and "c".

Each of the next q lines contains an integer i and character c $(1 \leq i \leq n)$, index and the value of the new item in the string, respectively. It is guaranteed that character's c value is "a", "b" or "c".

Output

For each query output the minimal number of characters that would have to be replaced so that the string doesn't contain "abc" as a substring.

Example

input
9 10 abcabcabc 1 a 1 b 2 c 3 a 4 b 5 c 8 a 9 b 1 c 4 a
output
3 2 2 2 1 2 1 1 1

```
0
```

Note

Let's consider the state of the string after each query:

1. $s =$ "abcabcabc". In this case 3 replacements can be performed to get, for instance, string $s =$ "bbcaccabb". This string does not contain "abc" as a substring.
2. $s =$ "bbcabcabc". In this case 2 replacements can be performed to get, for instance, string $s =$ "bbcbbcbbc". This string does not contain "abc" as a substring.
3. $s =$ "bccabcabc". In this case 2 replacements can be performed to get, for instance, string $s =$ "bccbbcbbc". This string does not contain "abc" as a substring.
4. $s =$ "bcaabcabc". In this case 2 replacements can be performed to get, for instance, string $s =$ "bcabbcbbc". This string does not contain "abc" as a substring.
5. $s =$ "bcabbcabc". In this case 1 replacements can be performed to get, for instance, string $s =$ "bcabbcabb". This string does not contain "abc" as a substring.
6. $s =$ "bcabccabc". In this case 2 replacements can be performed to get, for instance, string $s =$ "bcabbcabb". This string does not contain "abc" as a substring.
7. $s =$ "bcabccaac". In this case 1 replacements can be performed to get, for instance, string $s =$ "bcabbcaac". This string does not contain "abc" as a substring.
8. $s =$ "bcabccaab". In this case 1 replacements can be performed to get, for instance, string $s =$ "bcabbcaab". This string does not contain "abc" as a substring.
9. $s =$ "ccabccaab". In this case 1 replacements can be performed to get, for instance, string $s =$ "ccabbcaab". This string does not contain "abc" as a substring.
10. $s =$ "ccaaccaab". In this case the string does not contain "abc" as a substring and no replacements are needed.

C. Complex Market Analysis

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

While performing complex market analysis William encountered the following problem:

For a given array a of size n and a natural number e, calculate the number of pairs of natural numbers (i, k) which satisfy the following conditions:

- $1 \le i, k$
- $i + e \cdot k \le n$.
- Product $a_i \cdot a_{i+e} \cdot a_{i+2 \cdot e} \cdot \ldots \cdot a_{i+k \cdot e}$ is a prime number.

A prime number (or a prime) is a natural number greater than 1 that is not a product of two smaller natural numbers.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10\,000$). Description of the test cases follows.

The first line of each test case contains two integers n and e $(1 \le e \le n \le 2 \cdot 10^5)$, the number of items in the array and number e, respectively.

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

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

Output

For each test case output the answer in the following format:

Output one line containing the number of pairs of numbers (i, k) which satisfy the conditions.

Example

input
6 7 3 10 2 1 3 1 19 3 3 2 1 13 1 9 3 2 4 2 1 1 1 1 4 2 3 1 1 1 1 4 1 1 2 1 1 2 2 1 2

output
2 0 4 0 5 0

Note

In the first example test case two pairs satisfy the conditions:

1. $i = 2, k = 1$, for which the product is: $a_2 \cdot a_5 = 2$ which is a prime number.
2. $i = 3, k = 1$, for which the product is: $a_3 \cdot a_6 = 19$ which is a prime number.

In the second example test case there are no pairs that satisfy the conditions.

In the third example test case four pairs satisfy the conditions:

1. $i = 1, k = 1$, for which the product is: $a_1 \cdot a_4 = 2$ which is a prime number.
2. $i = 1, k = 2$, for which the product is: $a_1 \cdot a_4 \cdot a_7 = 2$ which is a prime number.
3. $i = 3, k = 1$, for which the product is: $a_3 \cdot a_6 = 2$ which is a prime number.
4. $i = 6, k = 1$, for which the product is: $a_6 \cdot a_9 = 2$ which is a prime number.

In the fourth example test case there are no pairs that satisfy the conditions.

In the fifth example test case five pairs satisfy the conditions:

1. $i = 1, k = 1$, for which the product is: $a_1 \cdot a_2 = 2$ which is a prime number.
2. $i = 1, k = 2$, for which the product is: $a_1 \cdot a_2 \cdot a_3 = 2$ which is a prime number.
3. $i = 1, k = 3$, for which the product is: $a_1 \cdot a_2 \cdot a_3 \cdot a_4 = 2$ which is a prime number.
4. $i = 2, k = 1$, for which the product is: $a_2 \cdot a_3 = 2$ which is a prime number.
5. $i = 2, k = 2$, for which the product is: $a_2 \cdot a_3 \cdot a_4 = 2$ which is a prime number.

In the sixth example test case there are no pairs that satisfy the conditions.

D. Social Network

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

William arrived at a conference dedicated to cryptocurrencies. Networking, meeting new people, and using friends' connections are essential to stay up to date with the latest news from the world of cryptocurrencies.

The conference has n participants, who are initially unfamiliar with each other. William can introduce any two people, a and b, who were not familiar before, to each other.

William has d conditions, i'th of which requires person x_i to have a connection to person y_i. Formally, two people x and y have a connection if there is such a chain $p_1 = x, p_2, p_3, \ldots, p_k = y$ for which for all i from 1 to $k - 1$ it's true that two people with numbers p_i and p_{i+1} know each other.

For every i ($1 \le i \le d$) William wants you to calculate the maximal number of acquaintances one person can have, assuming that William satisfied all conditions from 1 and up to and including i and performed **exactly** i introductions. The conditions are being checked after William performed i introductions. The answer for each i must be calculated independently. It means that when you compute an answer for i, you should assume that no two people have been introduced to each other yet.

Input

The first line contains two integers n and d ($2 \le n \le 10^3, 1 \le d \le n - 1$), the number of people, and number of conditions, respectively.

Each of the next d lines each contain two integers x_i and y_i ($1 \le x_i, y_i \le n, x_i \ne y_i$), the numbers of people which must have a connection according to condition i.

Output

Output d integers. ith number must equal the number of acquaintances the person with the maximal possible acquaintances will have, if William performed i introductions and satisfied the first i conditions.

Examples

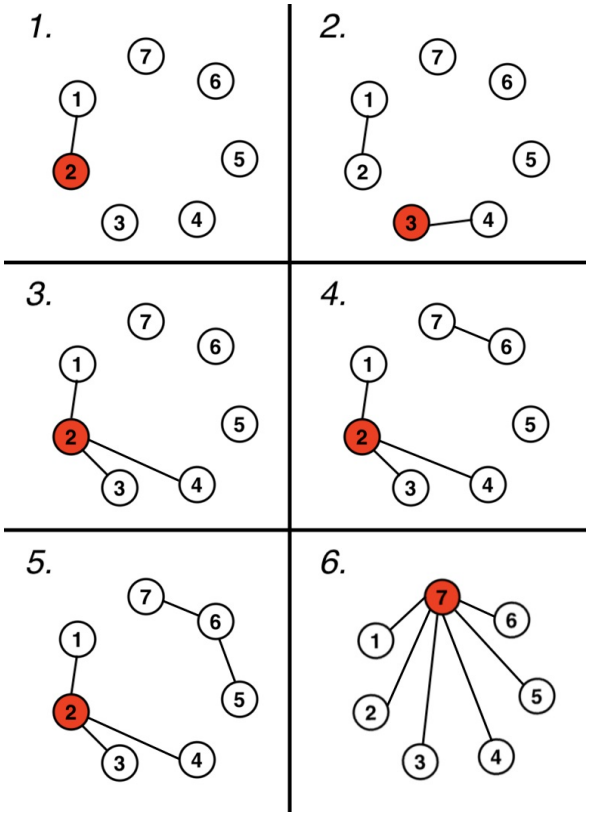
input
7 6 1 2 3 4 2 4 7 6 6 5 1 7
output
1 1 3 3 3 6

input
10 8 1 2 2 3 3 4 1 4 6 7 8 9 8 10 1 4
output
1 2 3 4 5 5 6 8

Note

The explanation for the first test case:

In this explanation, the circles and the numbers in them denote a person with the corresponding number. The line denotes that William introduced two connected people. The person marked with red has the most acquaintances. These are not the only correct ways to introduce people.



E. William The Oblivious

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

Before becoming a successful trader William got a university degree. During his education an interesting situation happened, after which William started to listen to homework assignments much more attentively. What follows is a formal description of the homework assignment as William heard it:

You are given a string s of length n only consisting of characters "a", "b" and "c". There are q queries of format (pos, c), meaning replacing the element of string s at position pos with character c. After each query you must output the minimal number of characters in the string, which have to be replaced, so that the string doesn't contain string "abc" as a **subsequence**. A valid replacement of a character is replacing it with "a", "b" or "c".

A string x is said to be a subsequence of string y if x can be obtained from y by deleting some characters without changing the ordering of the remaining characters.

Input

The first line contains two integers n and q $(1 \leq n, q \leq 10^5)$, the length of the string and the number of queries, respectively.

The second line contains the string s, consisting of characters "a", "b" and "c".

Each of the next q lines contains an integer i and character c $(1 \leq i \leq n)$, index and the value of the new item in the string, respectively. It is guaranteed that character's c value is "a", "b" or "c".

Output

For each query output the minimal number of characters that would have to be replaced so that the string doesn't contain "abc" as a subsequence.

Example

input
9 12 aaabccccc 4 a 4 b 2 b 5 a 1 b 6 b 5 c 2 a 1 a 5 a 6 b 7 b
output
0 1 2 2 1 2 1 2 2 2 2 2

Note

Let's consider the state of the string after each query:

1. s ="aaaaccccc". In this case the string does not contain "abc" as a subsequence and no replacements are needed.
2. s ="aaabccccc". In this case 1 replacements can be performed to get, for instance, string s ="aaaaccccc". This string does not contain "abc" as a subsequence.
3. s ="ababccccc". In this case 2 replacements can be performed to get, for instance, string s =aaaaccccc". This string does not contain "abc" as a subsequence.
4. s ="ababacccc". In this case 2 replacements can be performed to get, for instance, string s ="aaaaacccc". This string does not contain "abc" as a subsequence.
5. s ="bbabacccc". In this case 1 replacements can be performed to get, for instance, string s ="bbacacccc". This string does not contain "abc" as a subsequence.
6. s ="bbababccc". In this case 2 replacements can be performed to get, for instance, string s ="bbacacccc". This string does not contain "abc" as a subsequence.
7. s ="bbabcbccc". In this case 1 replacements can be performed to get, for instance, string s ="bbcbcbccc". This string does not contain "abc" as a subsequence.
8. s ="baabcbccc". In this case 2 replacements can be performed to get, for instance, string s ="bbbbcbccc". This string does not contain "abc" as a subsequence.
9. s ="aaabcbccc". In this case 2 replacements can be performed to get, for instance, string s ="aaaccccccc". This string does not contain "abc" as a subsequence.
10. s ="aaababccc". In this case 2 replacements can be performed to get, for instance, string s ="aaacacccc". This string does not contain "abc" as a subsequence.
11. s ="aaababccc". In this case 2 replacements can be performed to get, for instance, string s ="aaacacccc". This string does not contain "abc" as a subsequence.
12. s ="aaababbcc". In this case 2 replacements can be performed to get, for instance, string s ="aaababbbb". This string does not contain "abc" as a subsequence.

F. Interesting Sections

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

William has an array of non-negative numbers $a_1, a_2, \ldots, a_n$. He wants you to find out how many segments $l \leq r$ pass the check. The check is performed in the following manner:

1. The minimum and maximum numbers are found on the segment of the array starting at l and ending at r.
2. The check is considered to be passed if the binary representation of the minimum and maximum numbers have the same number of bits equal to 1.

Input

The first line contains a single integer n ($1 \leq n \leq 10^6$), the size of array a.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($0 \leq a_i \leq 10^{18}$), the contents of array a.

Output

Output a single number — the total number of segments that passed the check.

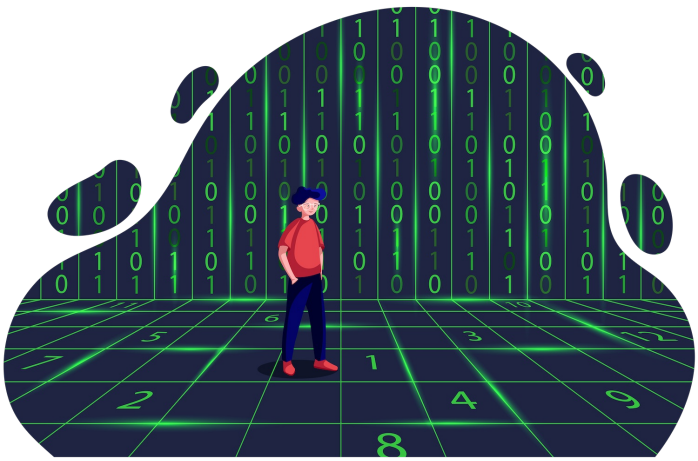
Examples

input
5 1 2 3 4 5
output
9

input
10 0 5 7 3 9 10 1 6 13 7
output
18

G. A Stroll Around the Matrix

time limit per test: 6 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output



William has two arrays of numbers $a_1, a_2, \ldots, a_n$ and $b_1, b_2, \ldots, b_m$. The arrays satisfy the conditions of being convex. Formally an array c of length k is considered convex if $c_i - c_{i-1} < c_{i+1} - c_i$ for all i from 2 to $k-1$ and $c_1 < c_2$.

Throughout William's life he observed q changes of two types happening to the arrays:

1. Add the arithmetic progression $d, d \cdot 2, d \cdot 3, \ldots, d \cdot k$ to the suffix of the array a of length k. The array after the change looks

like this: $[a_1, a_2, \ldots, a_{n-k}, a_{n-k+1} + d, a_{n-k+2} + d \cdot 2, \ldots, a_n + d \cdot k]$.
2. The same operation, but for array b.

After each change a matrix d is created from arrays a and b, of size $n \times m$, where $d_{i,j} = a_i + b_j$. William wants to get from cell $(1, 1)$ to cell (n, m) of this matrix. From cell (x, y) he can only move to cells $(x + 1, y)$ and $(x, y + 1)$. The length of a path is calculated as the sum of numbers in cells visited by William, including the first and the last cells.

After each change William wants you to help find out the minimal length of the path he could take.

Input

The first line contains three integers n, m and q ($2 \le n \le 100, 2 \le m \le 10^5$, $1 \le q \le 10^5$), the sizes of the arrays and the number of changes.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^{12}$), the contents of array a.

The third line contains m integers $b_1, b_2, \ldots, b_m$ ($1 \le b_i \le 10^{12}$), the contents of array b.

Each of the next q lines contains three integers $type$, k and d ($1 \le type \le 2$, if $type = 1$, then $1 \le k \le n$ otherwise $1 \le k \le m$, $1 \le d \le 10^3$).

Output

After each change, output one integer, the minimum length of the path in the constructed matrix.

Examples

input
5 3 4 1 2 4 7 11 5 7 10 1 3 2 2 2 5 1 5 4 2 1 7
output
98 128 219 229

input
5 6 7 4 9 22 118 226 7 94 238 395 565 738 2 1 95 1 4 54 1 2 5 1 2 87 2 6 62 2 1 143 1 1 77
output
3639 5122 5162 5617 7663 7806 7960

H. Pushing Robots

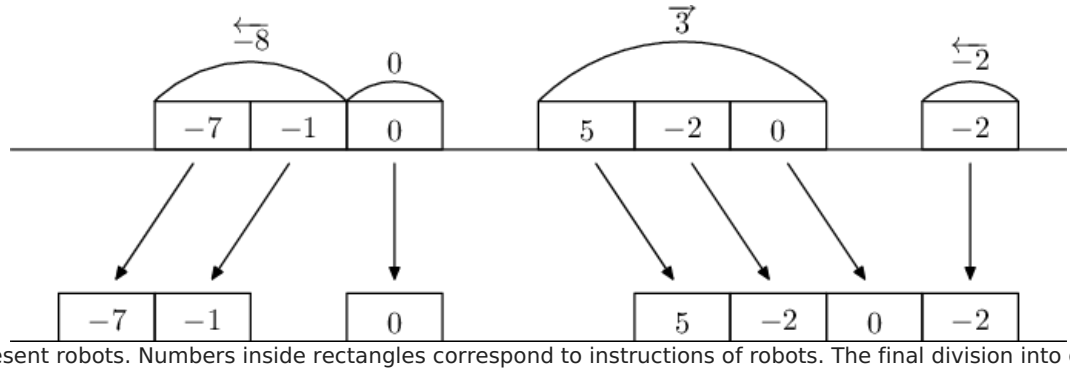
time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

There're n robots placed on a number line. Initially, i-th of them occupies unit segment $[x_i, x_i + 1]$. Each robot has a program, consisting of k instructions numbered from 1 to k. The robot performs instructions in a cycle. Each instruction is described by an integer number. Let's denote the number corresponding to the j-th instruction of the i-th robot as $f_{i,j}$.

Initial placement of robots corresponds to the moment of time 0. During one second from moment of time t ($0 \le t$) until $t + 1$ the following process occurs:

1. Each robot performs $(t \bmod k + 1)$-th instruction from its list of instructions. Robot number i takes number $F = f_{i,(t \bmod k+1)}$. If this number is negative (less than zero), the robot is trying to move to the left with force $|F|$. If the number is positive (more than zero), the robot is trying to move to the right with force F. Otherwise, the robot does nothing.
2. Let's imaginary divide robots into groups of consecutive, using the following algorithm:
   - Initially, each robot belongs to its own group.
   - Let's sum up numbers corresponding to the instructions of the robots from one group. Note that we are summing numbers without taking them by absolute value. Denote this sum as S. We say that the whole group moves together, and does it with force S by the same rules as a single robot. That is if S is negative, the group is trying to move to the left with force $|S|$. If S is positive, the group is trying to move to the right with force S. Otherwise, the group does nothing.
   - If one group is trying to move, and in the direction of movement touches another group, let's unite them. One group is touching another if their outermost robots occupy adjacent unit segments.
   - Continue this process until groups stop uniting.
3. Each robot moves by 1 in the direction of movement of its group or stays in place if its group isn't moving. But there's one exception.
4. The exception is if there're two groups of robots, divided by exactly one unit segment, such that the left group is trying to move to the right and the right group is trying to move to the left. Let's denote sum in the left group as S_l and sum in the right group as S_r. If $|S_l| \le |S_r|$ only the right group will move. Otherwise, only the left group will move.
5. Note that robots from one group don't glue together. They may separate in the future. The division into groups is imaginary and is needed only to understand how robots will move during one second $[t, t + 1]$.

An illustration of the process happening during one second:



Rectangles represent robots. Numbers inside rectangles correspond to instructions of robots. The final division into groups is marked with arcs. Below are the positions of the robots after moving. Only the left of the two rightmost groups moved. That's because these two groups tried to move towards each other, and were separated by exactly one unit segment.

Look at the examples for a better understanding of the process.

You need to answer several questions. What is the position of a_i-th robot at the moment of time t_i?

Input

The first line contains two integer numbers n and k — the number of robots and the number of instructions in the program of one robot ($1 \le n \le 100$, $1 \le k \le 50$).

The next line contains n integer numbers x_i — positions of robots at moment of time 0 ($-10^9 \le x_1 < x_2 < \cdots < x_n < 10^9$).

The next n lines contain descriptions of robots' programs. The i-th of these lines contains k integer numbers $f_{i,j}$ ($|f_{i,j}| \le 10^6$).

The next line contains a single integer number q — the number of questions you to answer ($1 \le q \le 1000$).

The next q lines contain two integer number a_i and t_i each, meaning that you should find a position of a_i-th robot at moment of time t_i ($1 \le a_i \le n, 0 \le t_i \le 10^{18}$).

Output

For every question output a single integer on the new line. Coordinate of the left border of unit segment occupied by the a_i-th robot at the moment of time t_i.

Examples

input
2 1 0 4 1 -1 8 1 0 2 0 1 1 2 1 1 2 2 2 1 3 2 3
output
0 4 1 3 1 2 1 2

input
2 1 0 4 2 -1 8 1 0 2 0 1 1 2 1 1 2 2 2 1 3 2 3
output
0 4 1 3 2 3 3 4

input
2 2 0 1 1 -1 -1 1 4 1 0 1 1 1 2 1 3
output
0 0 -1 0

input
1 3 0 3 -2 1 3 1 5 1 10 1 15

output

```
1
4
5
```

input

```
4 3
-8 -4 2 5
-1 3 0
1 -3 -4
2 -5 2
-1 -4 2
5
3 12
4 18
4 11
1 6
1 10
```

output

```
6
9
6
-8
-9
```