A. Lunar New Year and Cross Counting

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

Lunar New Year is approaching, and you bought a matrix with lots of "crosses".

This matrix M of size $n \times n$ contains only 'X' and '.' (without quotes). The element in the i-th row and the j-th column (i, j) is defined as $M(i, j)$, where $1 \leq i, j \leq n$. We define a *cross* appearing in the i-th row and the j-th column ($1 < i, j < n$) if and only if $M(i, j) = M(i - 1, j - 1) = M(i - 1, j + 1) = M(i + 1, j - 1) = M(i + 1, j + 1) =$ 'X'.

The following figure illustrates a cross appearing at position $(2, 2)$ in a 3×3 matrix.

X.X
.X.
X.X

Your task is to find out the number of *crosses* in the given matrix M. Two *crosses* are different if and only if they appear in different rows or columns.

Input

The first line contains only one positive integer n ($1 \leq n \leq 500$), denoting the size of the matrix M.

The following n lines illustrate the matrix M. Each line contains exactly n characters, each of them is 'X' or '.'. The j-th element in the i-th line represents $M(i, j)$, where $1 \leq i, j \leq n$.

Output

Output a single line containing only one integer number k — the number of *crosses* in the given matrix M.

Examples

input
5 ..... .XXX. .XXX. .XXX. .....
output
1

input
2 XX XX
output
0

input
6 ...... X.X.X. .X.X.X X.X.X. .X.X.X ......
output
4

Note

In the first sample, a *cross* appears at $(3, 3)$, so the answer is 1.

In the second sample, no *crosses* appear since $n < 3$, so the answer is 0.

In the third sample, *crosses* appear at $(3, 2)$, $(3, 4)$, $(4, 3)$, $(4, 5)$, so the answer is 4.

B. Lunar New Year and Food Ordering

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

Lunar New Year is approaching, and Bob is planning to go for a famous restaurant — "Alice's".

The restaurant "Alice's" serves n kinds of food. The cost for the i-th kind is always c_i. Initially, the restaurant has enough ingredients for serving exactly a_i dishes of the i-th kind. In the New Year's Eve, m customers will visit Alice's one after another and the j-th customer will order d_j dishes of the t_j-th kind of food. The $(i+1)$-st customer will only come after the i-th customer is completely served.

Suppose there are r_i dishes of the i-th kind remaining (initially $r_i = a_i$). When a customer orders 1 dish of the i-th kind, the following principles will be processed.

1. If $r_i > 0$, the customer will be served exactly 1 dish of the i-th kind. The cost for the dish is c_i. Meanwhile, r_i will be reduced by 1.
2. Otherwise, the customer will be served 1 dish of the **cheapest** available kind of food if there are any. If there are multiple cheapest kinds of food, the one with the smallest index among the cheapest will be served. The cost will be the cost for the dish served and the remain for the corresponding dish will be reduced by 1.
3. If there are no more dishes at all, the customer will leave angrily. Therefore, no matter how many dishes are served previously, the cost for the customer is 0.

If the customer doesn't leave after the d_j dishes are served, the cost for the customer will be the sum of the cost for these d_j dishes.

Please determine the total cost for each of the m customers.

Input

The first line contains two integers n and m ($1 \le n, m \le 10^5$), representing the number of different kinds of food and the number of customers, respectively.

The second line contains n positive integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^7$), where a_i denotes the initial remain of the i-th kind of dishes.

The third line contains n positive integers $c_1, c_2, \ldots, c_n$ ($1 \le c_i \le 10^6$), where c_i denotes the cost of one dish of the i-th kind.

The following m lines describe the orders of the m customers respectively. The j-th line contains two positive integers t_j and d_j ($1 \le t_j \le n$, $1 \le d_j \le 10^7$), representing the kind of food and the number of dishes the j-th customer orders, respectively.

Output

Print m lines. In the j-th line print the cost for the j-th customer.

Examples

input
8 5 8 6 2 1 4 5 7 5 6 3 3 2 6 2 3 2 2 8 1 4 4 7 3 4 6 10
output
22 24 14 10 39

input
6 6 6 6 6 6 6 6 6 66 666 6666 66666 666666 1 6 2 6 3 6 4 6 5 6 6 66
output
36 396 3996 39996 399996 0

input
6 6 6 6 6 6 6 6 6 66 666 6666 66666 666666 1 6 2 13 3 6 4 11 5 6 6 6

output
36 11058 99996 4333326 0 0

Note

In the first sample, 5 customers will be served as follows.

1. Customer 1 will be served 6 dishes of the 2-nd kind, 1 dish of the 4-th kind, and 1 dish of the 6-th kind. The cost is $6 \cdot 3 + 1 \cdot 2 + 1 \cdot 2 = 22$. The remain of the 8 kinds of food will be $\{8, 0, 2, 0, 4, 4, 7, 5\}$.
2. Customer 2 will be served 4 dishes of the 1-st kind. The cost is $4 \cdot 6 = 24$. The remain will be $\{4, 0, 2, 0, 4, 4, 7, 5\}$.
3. Customer 3 will be served 4 dishes of the 6-th kind, 3 dishes of the 8-th kind. The cost is $4 \cdot 2 + 3 \cdot 2 = 14$. The remain will be $\{4, 0, 2, 0, 4, 0, 7, 2\}$.
4. Customer 4 will be served 2 dishes of the 3-rd kind, 2 dishes of the 8-th kind. The cost is $2 \cdot 3 + 2 \cdot 2 = 10$. The remain will be $\{4, 0, 0, 0, 4, 0, 7, 0\}$.
5. Customer 5 will be served 7 dishes of the 7-th kind, 3 dishes of the 1-st kind. The cost is $7 \cdot 3 + 3 \cdot 6 = 39$. The remain will be $\{1, 0, 0, 0, 4, 0, 0, 0\}$.

In the second sample, each customer is served what they order **except** the last one, who leaves angrily without paying. For example, the second customer is served 6 dishes of the second kind, so the cost is $66 \cdot 6 = 396$.

In the third sample, some customers may not be served what they order. For example, the second customer is served 6 dishes of the second kind, 6 of the third and 1 of the fourth, so the cost is $66 \cdot 6 + 666 \cdot 6 + 6666 \cdot 1 = 11058$.

C. Lunar New Year and Number Division

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

Lunar New Year is approaching, and Bob is struggling with his homework – a number division problem.

There are n positive integers $a_1, a_2, \ldots, a_n$ on Bob's homework paper, where n is always an **even** number. Bob is asked to divide those numbers into groups, where each group must contain at least 2 numbers. Suppose the numbers are divided into m groups, and the sum of the numbers in the j-th group is s_j. Bob's aim is to minimize the sum of the square of s_j, that is

$$\sum_{j=1}^{m} s_j^2.$$

Bob is puzzled by this hard problem. Could you please help him solve it?

Input

The first line contains an **even** integer n ($2 \leq n \leq 3 \cdot 10^5$), denoting that there are n integers on Bob's homework paper.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 10^4$), describing the numbers you need to deal with.

Output

A single line containing one integer, denoting the minimum of the sum of the square of s_j, which is

$$\sum_{i=j}^{m} s_j^2,$$

where m is the number of groups.

Examples

input
4 8 5 2 3

output
164

input
6 1 1 1 2 2 2
output
27

Note

In the first sample, one of the optimal solutions is to divide those 4 numbers into 2 groups $\{2, 8\}, \{5, 3\}$. Thus the answer is $(2 + 8)^2 + (5 + 3)^2 = 164$.

In the second sample, one of the optimal solutions is to divide those 6 numbers into 3 groups $\{1, 2\}, \{1, 2\}, \{1, 2\}$. Thus the answer is $(1 + 2)^2 + (1 + 2)^2 + (1 + 2)^2 = 27$.

D. Lunar New Year and a Wander

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

Lunar New Year is approaching, and Bob decides to take a wander in a nearby park.

The park can be represented as a connected graph with n nodes and m bidirectional edges. Initially Bob is at the node 1 and he records 1 on his notebook. He can wander from one node to another through those bidirectional edges. Whenever he visits a node not recorded on his notebook, he records it. After he visits all nodes at least once, he stops wandering, thus finally a permutation of nodes $a_1, a_2, \ldots, a_n$ is recorded.

Wandering is a boring thing, but solving problems is fascinating. Bob wants to know the lexicographically smallest sequence of nodes he can record while wandering. Bob thinks this problem is trivial, and he wants you to solve it.

A sequence x is lexicographically smaller than a sequence y if and only if one of the following holds:

- x is a prefix of y, but $x \neq y$ (this is impossible in this problem as all considered sequences have the same length);
- in the first position where x and y differ, the sequence x has a smaller element than the corresponding element in y.

Input

The first line contains two positive integers n and m ($1 \leq n, m \leq 10^5$), denoting the number of nodes and edges, respectively.

The following m lines describe the bidirectional edges in the graph. The i-th of these lines contains two integers u_i and v_i ($1 \leq u_i, v_i \leq n$), representing the nodes the i-th edge connects.

Note that the graph can have multiple edges connecting the same two nodes and self-loops. It is guaranteed that the graph is connected.

Output

Output a line containing the lexicographically smallest sequence $a_1, a_2, \ldots, a_n$ Bob can record.

Examples

input
3 2 1 2 1 3
output
1 2 3

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

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

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

Note

In the first sample, Bob's optimal wandering path could be $1 \to 2 \to 1 \to 3$. Therefore, Bob will obtain the sequence $\{1, 2, 3\}$, which is the lexicographically smallest one.

In the second sample, Bob's optimal wandering path could be $1 \to 4 \to 3 \to 2 \to 3 \to 4 \to 1 \to 5$. Therefore, Bob will obtain the sequence $\{1, 4, 3, 2, 5\}$, which is the lexicographically smallest one.

E. Lunar New Year and Red Envelopes

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

Lunar New Year is approaching, and Bob is going to receive some red envelopes with countless money! But collecting money from red envelopes is a time-consuming process itself.

Let's describe this problem in a mathematical way. Consider a timeline from time 1 to n. The i-th red envelope will be available from time s_i to t_i, inclusive, and contain w_i coins. If Bob chooses to collect the coins in the i-th red envelope, he can do it only in an **integer** point of time between s_i and t_i, inclusive, and he can't collect any more envelopes until time d_i (inclusive) after that. Here $s_i \leq t_i \leq d_i$ holds.

Bob is a greedy man, he collects coins greedily — whenever he can collect coins at some integer time x, he collects the available red envelope with the maximum number of coins. If there are multiple envelopes with the same maximum number of coins, Bob would choose the one whose parameter d is the **largest**. If there are still multiple choices, Bob will choose one from them randomly.

However, Alice — his daughter — doesn't want her father to get too many coins. She could disturb Bob at no more than m integer time moments. If Alice decides to disturb Bob at time x, he could not do anything at time x and resumes his usual strategy at the time $x + 1$ (inclusive), which may lead to missing some red envelopes.

Calculate the minimum number of coins Bob would get if Alice disturbs him optimally.

Input

The first line contains three non-negative integers n, m and k ($1 \leq n \leq 10^5$, $0 \leq m \leq 200$, $1 \leq k \leq 10^5$), denoting the length of the timeline, the number of times Alice can disturb Bob and the total number of red envelopes, respectively.

The following k lines describe those k red envelopes. The i-th line contains four positive integers s_i, t_i, d_i and w_i ($1 \leq s_i \leq t_i \leq d_i \leq n$, $1 \leq w_i \leq 10^9$) — the time segment when the i-th envelope is available, the time moment Bob can continue collecting after collecting the i-th envelope, and the number of coins in this envelope, respectively.

Output

Output one integer — the minimum number of coins Bob would get if Alice disturbs him optimally.

Examples

input
5 0 2 1 3 4 5 2 5 5 8
output
13

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

input
12 2 6 1 5 5 4 4 6 6 2 3 8 8 3 2 9 9 5 6 10 10 7

8 12 12 9

output
11

Note

In the first sample, Alice has no chance to disturb Bob. Therefore Bob will collect the coins in the red envelopes at time 1 and 5, collecting 13 coins in total.

In the second sample, Alice should disturb Bob at time 1. Therefore Bob skips the first envelope, collects the second one and can not do anything after that. So the answer is 2.

F. Lunar New Year and a Recursive Sequence

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

Lunar New Year is approaching, and Bob received a gift from his friend recently — a recursive sequence! He loves this sequence very much and wants to play with it.

Let $f_1, f_2, \ldots, f_i, \ldots$ be an infinite sequence of positive integers. Bob knows that for $i > k$, f_i can be obtained by the following recursive equation:

$$f_i = \left(f_{i-1}^{b_1} \cdot f_{i-2}^{b_2} \cdot \cdots \cdot f_{i-k}^{b_k}\right) \bmod p,$$

which in short is

$$f_i = \left(\prod_{j=1}^{k} f_{i-j}^{b_j}\right) \bmod p,$$

where $p = 998\,244\,353$ (a widely-used prime), $b_1, b_2, \ldots, b_k$ are known integer constants, and $x \bmod y$ denotes the remainder of x divided by y.

Bob lost the values of $f_1, f_2, \ldots, f_k$, which is extremely troublesome – these are the basis of the sequence! Luckily, Bob remembers the first $k - 1$ elements of the sequence: $f_1 = f_2 = \ldots = f_{k-1} = 1$ and the n-th element: $f_n = m$. Please find any possible value of f_k. If no solution exists, just tell Bob that it is impossible to recover his favorite sequence, regardless of Bob's sadness.

Input

The first line contains a positive integer k ($1 \le k \le 100$), denoting the length of the sequence $b_1, b_2, \ldots, b_k$.

The second line contains k positive integers $b_1, b_2, \ldots, b_k$ ($1 \le b_i < p$).

The third line contains two positive integers n and m ($k < n \le 10^9$, $1 \le m < p$), which implies $f_n = m$.

Output

Output a possible value of f_k, where f_k is a positive integer satisfying $1 \le f_k < p$. If there are multiple answers, print any of them. If no such f_k makes $f_n = m$, output -1 instead.

It is easy to show that if there are some possible values of f_k, there must be at least one satisfying $1 \le f_k < p$.

Examples

input
3 2 3 5 4 16

output
4

input
5 4 7 1 5 6 7 14187219

output
6

input
8 2 3 5 6 1 7 9 10 23333 1

output

1

input
1 2 88888 66666
output
-1

input
3 998244352 998244352 998244352 4 2
output
-1

input
10 283 463 213 777 346 201 463 283 102 999 2333333 6263423
output
382480067

Note

In the first sample, we have $f_4 = f_3^2 \cdot f_2^3 \cdot f_1^5$. Therefore, applying $f_3 = 4$, we have $f_4 = 16$. Note that there can be multiple answers.

In the third sample, applying $f_7 = 1$ makes $f_{23333} = 1$.

In the fourth sample, no such f_1 makes $f_{88888} = 66666$. Therefore, we output -1 instead.