Codeforces Round #502 (in memory of Leopoldo Taravilse, Div. 1 + Div. 2)

A. The Rank

1 second, 256 megabytes

John Smith knows that his son, Thomas Smith, is among the best students in his class and even in his school. After the students of the school took the exams in English, German, Math, and History, a table of results was formed.

There are n students, each of them has a **unique** id (from 1 to n). Thomas's id is 1. Every student has four scores correspond to his or her English, German, Math, and History scores. The students are given in order of increasing of their ids.

In the table, the students will be sorted by **decreasing** the sum of their scores. So, a student with the largest sum will get the first place. If two or more students have the same sum, these students will be sorted by **increasing** their ids.

Please help John find out the rank of his son.

Input

The first line contains a single integer n ($1 \le n \le 1000$) — the number of students.

Each of the next n lines contains four integers a_i, b_i, c_i, and d_i ($0 \le a_i, b_i, c_i, d_i \le 100$) — the grades of the i-th student on English, German, Math, and History. The id of the i-th student is equal to i.

Output

Print the rank of Thomas Smith. Thomas's id is 1.

input
5 100 98 100 100 100 100 100 100 100 100 99 99 90 99 90 100 100 98 60 99
output
2

input
6 100 80 90 99 60 60 60 60 90 60 100 60 60 100 60 80 100 100 0 100 0 0 0 0
output
1

In the first sample, the students got total scores: 398, 400, 398, 379, and 357. Among the 5 students, Thomas and the third student have the second highest score, but Thomas has a smaller id, so his rank is 2.

In the second sample, the students got total scores: 369, 240, 310, 300, 300, and 0. Among the 6 students, Thomas got the highest score, so his rank is 1.

B. The Bits

2 seconds, 256 megabytes

Rudolf is on his way to the castle. Before getting into the castle, the security staff asked him a question:

Given two binary numbers a and b of length n. How many different ways of swapping two digits in a (only in a, not b) so that bitwise OR of these two numbers will be changed? In other words, let c be the bitwise OR of a and b, you need to find the number of ways of swapping two bits in a so that bitwise OR will not be equal to c.

Note that binary numbers can contain leading zeros so that length of each number is exactly n.

Bitwise OR is a binary operation. A result is a binary number which contains a one in each digit if there is a one in at least one of the two numbers. For example, 01010_2 OR 10011_2 = 11011_2.

Well, to your surprise, you are not Rudolf, and you don't need to help him . . . You are the security staff! Please find the number of ways of swapping two bits in a so that bitwise OR will be changed.

Input

The first line contains one integer n ($2 \le n \le 10^5$) — the number of bits in each number.

The second line contains a binary number a of length n.

The third line contains a binary number b of length n.

Output

Print the number of ways to swap two bits in a so that bitwise OR will be changed.

input
5 01011 11001
output
4

input
6 011000 010011
output
6

In the first sample, you can swap bits that have indexes $(1, 4)$, $(2, 3)$, $(3, 4)$, and $(3, 5)$.

In the second example, you can swap bits that have indexes $(1, 2)$, $(1, 3)$, $(2, 4)$, $(3, 4)$, $(3, 5)$, and $(3, 6)$.

C. The Phone Number

1 second, 256 megabytes

Mrs. Smith is trying to contact her husband, John Smith, but she forgot the secret phone number!

The only thing Mrs. Smith remembered was that any permutation of n can be a secret phone number. Only those permutations that minimize secret value might be the phone of her husband.

The sequence of n integers is called a permutation if it contains all integers from 1 to n exactly once.

The secret value of a phone number is defined as the sum of the length of the longest increasing subsequence (LIS) and length of the longest decreasing subsequence (LDS).

A subsequence $a_{i_1}, a_{i_2}, \ldots, a_{i_k}$ where $1 \le i_1 < i_2 < \ldots < i_k \le n$ is called increasing if $a_{i_1} < a_{i_2} < a_{i_3} < \ldots < a_{i_k}$. If $a_{i_1} > a_{i_2} > a_{i_3} > \ldots > a_{i_k}$, a subsequence is called decreasing. An increasing/decreasing subsequence is called longest if it has maximum length among all increasing/decreasing subsequences.

For example, if there is a permutation $[6, 4, 1, 7, 2, 3, 5]$, LIS of this permutation will be $[1, 2, 3, 5]$, so the length of LIS is equal to 4. LDS can be $[6, 4, 1]$, $[6, 4, 2]$, or $[6, 4, 3]$, so the length of LDS is 3.

Note, the lengths of LIS and LDS can be different.

So please help Mrs. Smith to find a permutation that gives a minimum sum of lengths of LIS and LDS.

Input
The only line contains one integer n ($1 \le n \le 10^5$) — the length of permutation that you need to build.

Output
Print a permutation that gives a minimum sum of lengths of LIS and LDS.

If there are multiple answers, print any.

input
4
output
3 4 1 2

input
2
output
2 1

In the first sample, you can build a permutation $[3, 4, 1, 2]$. LIS is $[3, 4]$ (or $[1, 2]$), so the length of LIS is equal to 2. LDS can be ony of $[3, 1]$, $[4, 2]$, $[3, 2]$, or $[4, 1]$. The length of LDS is also equal to 2. The sum is equal to 4. Note that $[3, 4, 1, 2]$ is **not** the only permutation that is valid.

In the second sample, you can build a permutation $[2, 1]$. LIS is $[1]$ (or $[2]$), so the length of LIS is equal to 1. LDS is $[2, 1]$, so the length of LDS is equal to 2. The sum is equal to 3. Note that permutation $[1, 2]$ is also valid.

D. The Wu

2 seconds, 256 megabytes

Childan is making up a legendary story and trying to sell his forgery — a necklace with a strong sense of "*Wu*" to the Kasouras. But Mr. Kasoura is challenging the truth of Childan's story. So he is going to ask a few questions about Childan's so-called "personal treasure" necklace.

This "personal treasure" is a multiset S of m "*01-strings*".

A "*01-string*" is a string that contains n characters "0" and "1". For example, if $n = 4$, strings "0110", "0000", and "1110" are "*01-strings*", but "00110" (there are 5 characters, not 4) and "zero" (unallowed characters) are not.

Note that the multiset S can contain equal elements.

Frequently, Mr. Kasoura will provide a "*01-string*" t and ask Childan how many strings s are in the multiset S such that the "*Wu*" value of the pair (s, t) is **not greater** than k.

Mrs. Kasoura and Mr. Kasoura think that if $s_i = t_i$ ($1 \le i \le n$) then the "*Wu*" value of the character pair equals to w_i, otherwise 0. The "*Wu*" value of the "*01-string*" pair is the sum of the "*Wu*" values of every character pair. Note that the length of every "*01-string*" is equal to n.

For example, if $w = [4, 5, 3, 6]$, "*Wu*" of ("1001", "1100") is 7 because these strings have equal characters only on the first and third positions, so $w_1 + w_3 = 4 + 3 = 7$.

You need to help Childan to answer Mr. Kasoura's queries. That is to find the number of strings in the multiset S such that the "*Wu*" value of the pair is not greater than k.

Input
The first line contains three integers n, m, and q ($1 \le n \le 12$, $1 \le q, m \le 5 \cdot 10^5$) — the length of the "*01-strings*", the size of the multiset S, and the number of queries.

The second line contains n integers $w_1, w_2, \ldots, w_n$ ($0 \le w_i \le 100$) — the value of the i-th caracter.

Each of the next m lines contains the "*01-string*" s of length n — the string in the multiset S.

Each of the next q lines contains the "*01-string*" t of length n and integer k ($0 \le k \le 100$) — the query.

Output
For each query, print the answer for this query.

input
2 4 5 40 20 01 01 10 11 00 20 00 40 11 20 11 40 11 60
output
2 4 2 3 4

input
1 2 4 100 0 1 0 0 0 100 1 0 1 100
output
1 2 1 2

In the first example, we can get:

"*Wu*" of ("01", "00") is 40.

"*Wu*" of ("10", "00") is 20.

"*Wu*" of ("11", "00") is 0.

"*Wu*" of ("01", "11") is 20.

"*Wu*" of ("10", "11") is 40.

"*Wu*" of ("11", "11") is 60.

In the first query, pairs ("11", "00") and ("10", "00") satisfy the condition since their "*Wu*" is not greater than 20.

In the second query, all strings satisfy the condition.

In the third query, pairs ("01", "11") and ("01", "11") satisfy the condition. Note that since there are two "01" strings in the multiset, the answer is 2, not 1.

In the fourth query, since k was increased, pair ("10", "11") satisfies the condition too.

In the fifth query, since k was increased, pair ("11", "11") satisfies the condition too.

E. The Supersonic Rocket

1 second, 256 megabytes

After the war, the supersonic rocket became the most common public transportation.

Each supersonic rocket consists of **two** "*engines*". Each engine is a set of "*power sources*". The first engine has n power sources, and the second one has m power sources. A power source can be described as a point (x_i, y_i) on a 2-D plane. All points in each engine are different.

You can manipulate each engine **separately**. There are two operations that you can do with each engine. You can do each operation as many times as you want.

1. For every power source as a whole in that engine: (x_i, y_i) becomes $(x_i + a, y_i + b)$, a and b can be any real numbers. In other words, all power sources will be shifted.
2. For every power source as a whole in that engine: (x_i, y_i) becomes $(x_i \cos\theta - y_i \sin\theta, x_i \sin\theta + y_i \cos\theta)$, θ can be any real number. In other words, all power sources will be rotated.

The engines work as follows: after the two engines are powered, their power sources are being combined (here power sources of different engines may coincide). If two power sources $A(x_a, y_a)$ and $B(x_b, y_b)$ exist, then for all real number k that $0 < k < 1$, a new power source will be created $C_k(kx_a + (1-k)x_b, ky_a + (1-k)y_b)$. **Then, this procedure will be repeated again with all new and old power sources**. After that, the "*power field*" from all power sources will be generated (can be considered as an infinite set of all power sources occurred).

A supersonic rocket is "*safe*" if and only if after you manipulate the engines, destroying any power source and then power the engine, the power field generated won't be changed (comparing to the situation where no power source erased). Two power fields are considered different if and only if any power source in one field belongs to the other one as well.

Given a supersonic rocket, check whether it is safe or not.

Input

The first line contains two integers n, m ($3 \le n, m \le 10^5$) — the number of power sources in each engine.

Each of the next n lines contains two integers x_i and y_i ($0 \le x_i, y_i \le 10^8$) — the coordinates of the i-th power source in the first engine.

Each of the next m lines contains two integers x_i and y_i ($0 \le x_i, y_i \le 10^8$) — the coordinates of the i-th power source in the second engine.

It is guaranteed that there are no two or more power sources that are located in the same point in each engine.

Output

Print "YES" if the supersonic rocket is safe, otherwise "NO".

You can print each letter in an arbitrary case (upper or lower).

input

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

output

```
YES
```

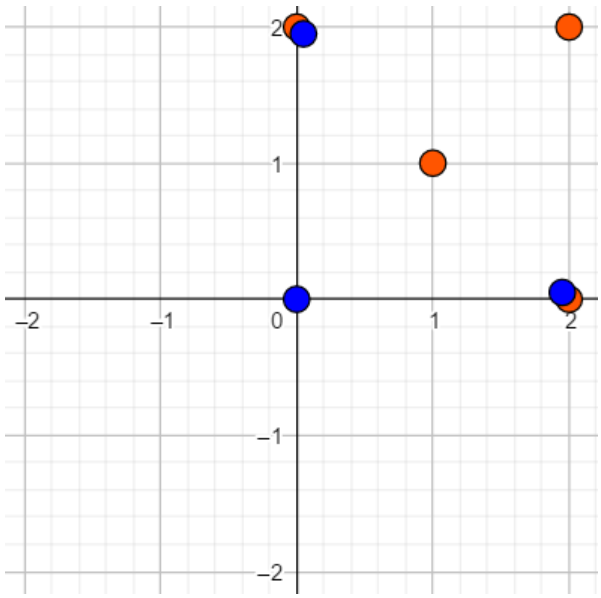
input

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

output

```
NO
```

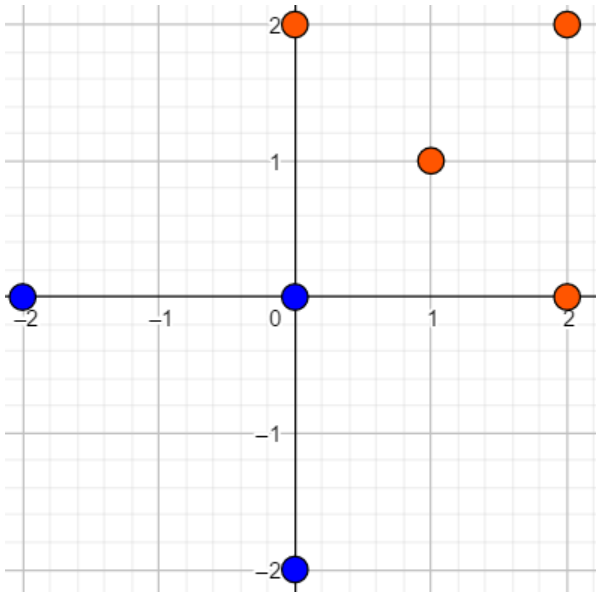
The first sample:



Those near pairs of blue and orange points actually coincide.

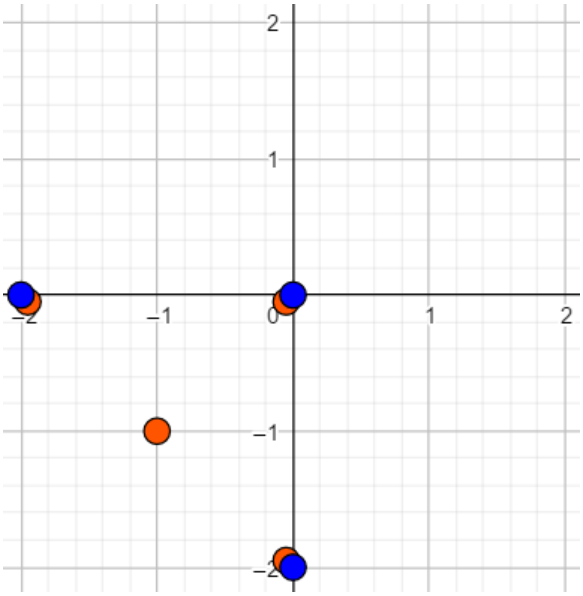
First, manipulate the first engine: use the second operation with $\theta = \pi$ (to rotate all power sources 180 degrees).

The power sources in the first engine become $(0, 0)$, $(0, -2)$, and $(-2, 0)$.



Second, manipulate the second engine: use the first operation with $a = b = -2$.

The power sources in the second engine become $(-2, 0)$, $(0, 0)$, $(0, -2)$, and $(-1, -1)$.



You can examine that destroying any point, the power field formed by the two engines are always the solid triangle $(0, 0)$, $(-2, 0)$, $(0, -2)$.

In the second sample, no matter how you manipulate the engines, there always exists a power source in the second engine that power field will shrink if you destroy it.

F. The Neutral Zone

5 seconds, 16 megabytes

Notice: unusual memory limit!

After the war, destroyed cities in the neutral zone were restored. And children went back to school.

The war changed the world, as well as education. In those hard days, a new math concept was created.

As we all know, logarithm function can be described as:

$$\log(p_1^{a1} p_2^{a2} \ldots p_k^{a2}) = a_1 \log p_1 + a_2 \log p_2 + \ldots + a_k \log p_k$$

Where $p_1^{a1} p_2^{a2} \ldots p_k^{a2}$ is the prime factorization of a integer. A problem is that the function uses itself in the definition. That is why it is hard to calculate.

So, the mathematicians from the neutral zone invented this:

$$\text{exlog}_f(p_1^{a1} p_2^{a2} \ldots p_k^{a2}) = a_1 f(p_1) + a_2 f(p_2) + \ldots + a_k f(p_k)$$

Notice that $\text{exlog}_f(1)$ is always equal to 0.

This concept for any function f was too hard for children. So teachers told them that f can only be a polynomial of degree no more than 3 in daily uses (i.e., $f(x) = Ax^3 + Bx^2 + Cx + D$).

"Class is over! Don't forget to do your homework!" Here it is:

$$\sum_{i=1}^{n} \text{exlog}_f(i)$$

Help children to do their homework. Since the value can be very big, you need to find the answer modulo 2^{32}.

Input

The only line contains five integers n, A, B, C, and D ($1 \le n \le 3 \cdot 10^8$, $0 \le A, B, C, D \le 10^6$).

Output

Print the answer modulo 2^{32}.

input
12 0 0 1 0

i
4 1
out
136

In the firs

$\text{exlog}_f(1)$

$\text{exlog}_f(2)$ =

$\text{exlog}_f(3)$ =

$\text{exlog}_f(4) = 2$

$\text{exlog}_f(5) = 5$

$\text{exlog}_f(6) = 2 + 3$

$\text{exlog}_f(7) = 7$

$\text{exlog}_f(8) = 2 + 2 +$

$\text{exlog}_f(9) = 3 + 3 = 6$

$\text{exlog}_f(10) = 2 + 5 = 7$

$\text{exlog}_f(11) = 11$

$\text{exlog}_f(12) = 2 + 2 + 3 = 7$

$\sum_{i=1}^{12} \text{exlog}_f(i) = 63$

In the second sample:

$\text{exlog}_f(1) = 0$

$\text{exlog}_f(2) = (1 \times 2^3 + 2 \times 2^2 + 3$

$\text{exlog}_f(3) = (1 \times 3^3 + 2 \times 3^2 + 3 \times$

$\text{exlog}_f(4) = 2 \times \text{exlog}_f(2) = 52$

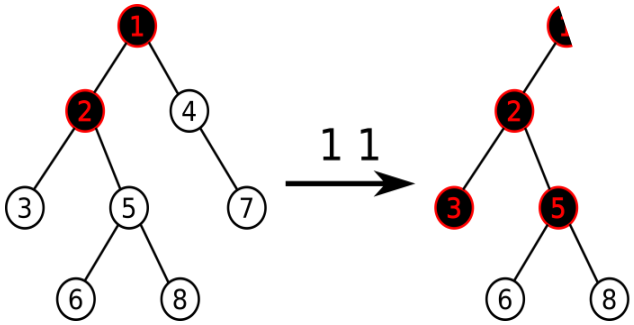
$\sum_{i=1}^{4} \text{exlog}_f(i) = 0 + 26 + 58 + 52 = 1$

G. The Tree

3 seconds, 256 megaby

Abendsen assigned a mission to Juliana. In this m
rooted tree with n vertices. Vertex number 1 is the
vertex can be either black or white. At first, all vertic
asked to process q queries. Each query is one of thre

1. If vertex v is white, mark it as black; otherwise, per
on all direct sons of v instead.
2. Mark all vertices in the subtree of v (including v) as w
3. Find the color of the i-th vertex.



An example of operation "1 1" (corresponds to the first example test). The vertices and 2 are already black, so the operation goes to their sons instead.

Can you help Juliana to process all these queries?

Input

The first line contains two integers n and q ($2 \le n \le 10^5$, $1 \le q \le 10^5$) — the number of vertices and the number of queries.

The second line contains $n-1$ integers $p_2, p_3, \ldots, p_n$ ($1 \le p_i < i$), where p_i means that there is an edge between vertices i and p_i.

Each of the next q lines contains two integers t_i and v_i ($1 \le t_i \le 3$, $1 \le v_i \le n$) — the type of the i-th query and the vertex of the i-th query.

It is guaranteed that the given graph is a tree.

Output

For each query of type 3, print "`black`" if the vertex is black; otherwise, print "`white`".

input

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

output

```
black
white
black
white
black
white
```

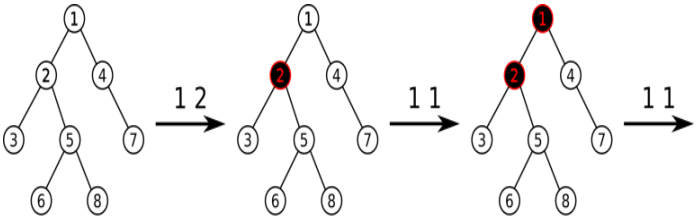
input

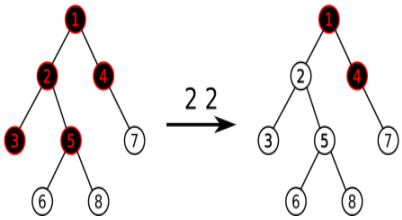
```
8 11
1 1 2 3 3 6 6
1 1
1 1
1 3
3 2
3 4
3 6
3 7
2 3
1 6
3 7
3 6
```

output

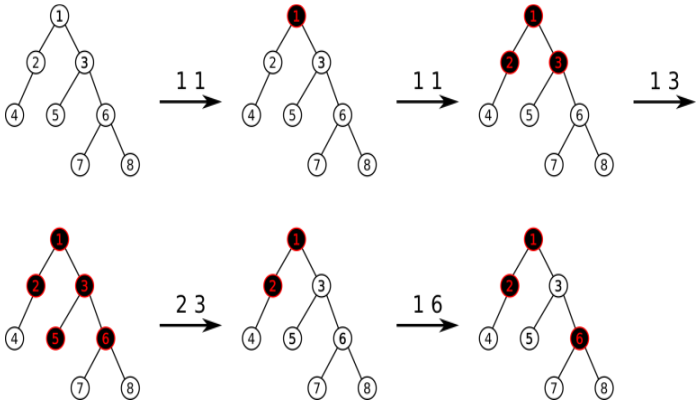
```
black
white
black
white
white
black
```

The first example is shown on the picture below.





The second example is shown on the picture below.



H. The Films

5 seconds, 512 megabytes

In "`The Man in the High Castle`" world, there are m different film endings.

Abendsen owns a storage and a shelf. At first, he has n ordered films on the shelf. In the i-th month he will do:

1. Empty the storage.
2. Put $k_i \cdot m$ films into the storage, k_i films for each ending.
3. He will think about a question: if he puts all the films from the shelf into the storage, then randomly picks n films (from all the films in the storage) and rearranges them on the shelf, what is the probability that sequence of endings in $[l_i, r_i]$ on the shelf will not be changed? Notice, he just thinks about this question, so the shelf will not actually be changed.

Answer all Abendsen's questions.

Let the probability be fraction P_i. Let's say that the total number of ways to take n films from the storage for i-th month is A_i, so $P_i \cdot A_i$ is always an integer. Print for each month $P_i \cdot A_i \pmod{998244353}$.

998244353 is a prime number and it is equal to $119 \cdot 2^{23} + 1$.

It is guaranteed that there will be only no more than 100 different k values.

Input

The first line contains three integers n, m, and q ($1 \le n, m, q \le 10^5$, $n + q \le 10^5$) — the number of films on the shelf initially, the number of endings, and the number of months.

The second line contains n integers $e_1, e_2, \ldots, e_n$ ($1 \le e_i \le m$) — the ending of the i-th film on the shelf.

Each of the next q lines contains three integers l_i, r_i, and k_i ($1 \le l_i \le r_i \le n, 0 \le k_i \le 10^5$) — the i-th query.

It is guaranteed that there will be only no more than 100 different k values.

Output

Print the answer for each question in a separate line.

input

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

output

```
6
26730
12150
4860
```

input

```
5 5 3
1 2 3 4 5
1 2 100000
1 4 4
3 5 5
```

output

```
494942218
13125
151632
```

In the first sample in the second query, after adding $2 \cdot m$ films into the storage, the storage will look like this: $\{1, 1, 1, 2, 2, 2, 3, 3, 3, 4, 4, 4, 4, 4\}$.

There are 26730 total ways of choosing the films so that $e_l, e_{l+1}, \ldots, e_r$ will not be changed, for example, $[1, 2, 3, 2, 2]$ and $[1, 2, 3, 4, 3]$ are such ways.

There are 2162160 total ways of choosing the films, so you're asked to print $(\frac{26730}{2162160} \cdot 2162160) \mod 998244353 = 26730$.