A. Regular Bracket Sequences

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

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

You are given an integer n. Your goal is to construct and print **exactly** n different regular bracket sequences of length $2n$.

Input

The first line contains one integer t ($1 \le t \le 50$) — the number of test cases.

Each test case consists of one line containing one integer n ($1 \le n \le 50$).

Output

For each test case, print n lines, each containing a regular bracket sequence of length **exactly** $2n$. All bracket sequences you output for a testcase should be different (though they may repeat in different test cases). If there are multiple answers, print any of them. It can be shown that it's always possible.

Example

input

```
3
3
1
3
```

output

```
()()()
((()))
(()())
()
((()))
(())()
()(())
```

B. Combinatorics Homework

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

You are given four integer values a, b, c and m.

Check if there exists a string that contains:

- a letters 'A';
- b letters 'B';
- c letters 'C';
- no other letters;
- exactly m pairs of adjacent equal letters (exactly m such positions i that the i-th letter is equal to the $(i+1)$-th one).

Input

The first line contains a single integer t ($1 \le t \le 10^4$) — the number of testcases.

Each of the next t lines contains the description of the testcase — four integers a, b, c and m ($1 \le a, b, c \le 10^8$; $0 \le m \le 10^8$).

Output

For each testcase print "YES" if there exists a string that satisfies all the requirements. Print "NO" if there are no such strings.

You may print every letter in any case you want (so, for example, the strings yEs, yes, Yes and YES will all be recognized as positive answer).

Example

input
3 2 2 1 0 1 1 1 1 1 2 3 2

output
YES NO YES

Note

In the first testcase strings "ABCAB" or "BCABA" satisfy the requirements. There exist other possible strings.

In the second testcase there's no way to put adjacent equal letters if there's no letter that appears at least twice.

In the third testcase string "CABBCC" satisfies the requirements. There exist other possible strings.

C. Slay the Dragon

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

Recently, Petya learned about a new game "Slay the Dragon". As the name suggests, the player will have to fight with dragons. To defeat a dragon, you have to kill it and defend your castle. To do this, the player has a squad of n heroes, the strength of the i-th hero is equal to a_i.

According to the rules of the game, exactly one hero should go kill the dragon, all the others will defend the castle. If the dragon's defense is equal to x, then you have to send a hero with a strength of at least x to kill it. If the dragon's attack power is y, then the total strength of the heroes defending the castle should be at least y.

The player can increase the strength of any hero by 1 for one gold coin. This operation can be done any number of times.

There are m dragons in the game, the i-th of them has defense equal to x_i and attack power equal to y_i. Petya was wondering what is the minimum number of coins he needs to spend to defeat the i-th dragon.

Note that the task is solved **independently for each dragon** (improvements are not saved).

Input

The first line contains a single integer n ($2 \le n \le 2 \cdot 10^5$) — number of heroes.

The second line contains n integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le 10^{12}$), where a_i is the strength of the i-th hero.

The third line contains a single integer m ($1 \le m \le 2 \cdot 10^5$) — the number of dragons.

The next m lines contain two integers each, x_i and y_i ($1 \le x_i \le 10^{12}; 1 \le y_i \le 10^{18}$) — defense and attack power of the i-th dragon.

Output

Print m lines, i-th of which contains a single integer — the minimum number of coins that should be spent to defeat the i-th dragon.

Example

input
4 3 6 2 3 5 3 12 7 9 4 14 1 10 8 7

output
1 2 4 0 2

Note

To defeat the first dragon, you can increase the strength of the third hero by 1, then the strength of the heroes will be equal to $[3, 6, 3, 3]$. To kill the dragon, you can choose the first hero.

To defeat the second dragon, you can increase the forces of the second and third heroes by 1, then the strength of the heroes will be equal to $[3, 7, 3, 3]$. To kill the dragon, you can choose a second hero.

To defeat the third dragon, you can increase the strength of all the heroes by 1, then the strength of the heroes will be equal to

$[4, 7, 3, 4]$. To kill the dragon, you can choose a fourth hero.

To defeat the fourth dragon, you don't need to improve the heroes and choose a third hero to kill the dragon.

To defeat the fifth dragon, you can increase the strength of the second hero by 2, then the strength of the heroes will be equal to $[3, 8, 2, 3]$. To kill the dragon, you can choose a second hero.

D. The Strongest Build

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

Ivan is playing yet another roguelike computer game. He controls a single hero in the game. The hero has n equipment slots. There is a list of c_i items for the i-th slot, the j-th of them increases the hero strength by $a_{i,j}$. The items for each slot are pairwise distinct and are listed in the increasing order of their strength increase. So, $a_{i,1} < a_{i,2} < \dots < a_{i,c_i}$.

For each slot Ivan chooses exactly one item. Let the chosen item for the i-th slot be the b_i-th item in the corresponding list. The sequence of choices $[b_1, b_2, \dots, b_n]$ is called *a build*.

The strength of a build is the sum of the strength increases of the items in it. Some builds are banned from the game. There is a list of m pairwise distinct banned builds. It's guaranteed that there's at least one build that's not banned.

What is the build with the maximum strength that is not banned from the game? If there are multiple builds with maximum strength, print any of them.

Input

The first line contains a single integer n ($1 \le n \le 10$) — the number of equipment slots.

The i-th of the next n lines contains the description of the items for the i-th slot. First, one integer c_i ($1 \le c_i \le 2 \cdot 10^5$) — the number of items for the i-th slot. Then c_i integers $a_{i,1}, a_{i,2}, \dots, a_{i,c_i}$ ($1 \le a_{i,1} < a_{i,2} < \dots < a_{i,c_i} \le 10^8$).

The sum of c_i doesn't exceed $2 \cdot 10^5$.

The next line contains a single integer m ($0 \le m \le 10^5$) — the number of banned builds.

Each of the next m lines contains a description of a banned build — a sequence of n integers $b_1, b_2, \dots, b_n$ ($1 \le b_i \le c_i$).

The builds are pairwise distinct, and there's at least one build that's not banned.

Output

Print the build with the maximum strength that is not banned from the game. If there are multiple builds with maximum strength, print any of them.

Examples

input

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

output

```
2 2 3
```

input

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

output

```
1 2 3
```

input

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

output

```
3 2 2
```

input

```
4
1 10
1 4
1 7
1 3
0
```

output

```
1 1 1 1
```

E. Coloring

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

A matrix of size $n \times m$, such that each cell of it contains either 0 or 1, is considered *beautiful* if the sum in every contiguous submatrix of size 2×2 is exactly 2, i. e. every "square" of size 2×2 contains exactly two 1's and exactly two 0's.

You are given a matrix of size $n \times m$. Initially each cell of this matrix is empty. Let's denote the cell on the intersection of the x-th row and the y-th column as (x, y). You have to process the queries of three types:

- x y -1 — clear the cell (x, y), if there was a number in it;
- x y 0 — write the number 0 in the cell (x, y), **overwriting the number that was there previously (if any)**;
- x y 1 — write the number 1 in the cell (x, y), **overwriting the number that was there previously (if any)**.

After each query, print the number of ways to fill the empty cells of the matrix so that the resulting matrix is *beautiful*. Since the answers can be large, print them modulo 998244353.

Input

The first line contains three integers n, m and k ($2 \le n, m \le 10^6$; $1 \le k \le 3 \cdot 10^5$) — the number of rows in the matrix, the number of columns, and the number of queries, respectively.

Then k lines follow, the i-th of them contains three integers x_i, y_i, t_i ($1 \le x_i \le n$; $1 \le y_i \le m$; $-1 \le t_i \le 1$) — the parameters for the i-th query.

Output

For each query, print one integer — the number of ways to fill the empty cells of the matrix after the respective query, taken modulo 998244353.

Example

input

```
2 2 7
1 1 1
1 2 1
2 1 1
1 1 0
1 2 -1
2 1 -1
1 1 -1
```

output

```
3
1
0
1
2
3
6
```

F. Occurrences

time limit per test: 7 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

A subarray of array a from index l to the index r is the array $[a_l, a_{l+1}, \dots, a_r]$. The number of occurrences of the array b in the array a is the number of subarrays of a such that they are equal to b.

You are given n arrays $A_1, A_2, \dots, A_n$; the elements of these arrays are integers from 1 to k. You have to build an array a consisting of m integers from 1 to k in such a way that, for **every** given subarray A_i, the number of occurrences of A_i in the array a

is **not less** than the number of occurrences of each non-empty subarray of A_i in a. Note that if A_i doesn't occur in a, and no subarray of A_i occurs in a, this condition is still met for A_i.

Your task is to calculate the number of different arrays a you can build, and print it modulo 998244353.

Input

The first line contains three integers n, m and k ($1 \le n, m, k \le 3 \cdot 10^5$) — the number of the given arrays, the desired length of the array a, and the upper bound on the values in the arrays.

Then n lines follow. The i-th line represents the array A_i. The first integer in the i-th line is c_i ($1 \le c_i \le m$) — the number of elements in A_i; then, c_i integers from 1 to k follow — the elements of the array A_i.

Additional constraint on the input: $\sum\limits_{i=1}^{n} c_i \le 3 \cdot 10^5$; i. e., the number of elements in the given arrays in total does not exceed $3 \cdot 10^5$.

Output

Print one integer — the number of different arrays a you can build, taken modulo 998244353.

Examples

input

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

output

```
5
```

input

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

output

```
0
```

input

```
1 42 1337
2 13 31
```

output

```
721234447
```