A. Circle Coloring

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

You are given three sequences: $a_1, a_2, \ldots, a_n$; $b_1, b_2, \ldots, b_n$; $c_1, c_2, \ldots, c_n$.

For each i, $a_i \neq b_i$, $a_i \neq c_i$, $b_i \neq c_i$.

Find a sequence $p_1, p_2, \ldots, p_n$, that satisfy the following conditions:

- $p_i \in \{a_i, b_i, c_i\}$
- $p_i \neq p_{(i \bmod n)+1}$.

In other words, for each element, you need to choose one of the three possible values, such that no two adjacent elements (where we consider elements $i, i+1$ adjacent for $i < n$ and also elements 1 and n) will have equal value.

It can be proved that in the given constraints solution always exists. You don't need to minimize/maximize anything, you need to find any proper sequence.

Input

The first line of input contains one integer t ($1 \leq t \leq 100$): the number of test cases.

The first line of each test case contains one integer n ($3 \leq n \leq 100$): the number of elements in the given sequences.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 100$).

The third line contains n integers $b_1, b_2, \ldots, b_n$ ($1 \leq b_i \leq 100$).

The fourth line contains n integers $c_1, c_2, \ldots, c_n$ ($1 \leq c_i \leq 100$).

It is guaranteed that $a_i \neq b_i$, $a_i \neq c_i$, $b_i \neq c_i$ for all i.

Output

For each test case, print n integers: $p_1, p_2, \ldots, p_n$ ($p_i \in \{a_i, b_i, c_i\}$, $p_i \neq p_{i \bmod n+1}$).

If there are several solutions, you can print any.

Example

input

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

output

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

Note

In the first test case $p = [1, 2, 3]$.

It is a correct answer, because:

- $p_1 = 1 = a_1$, $p_2 = 2 = b_2$, $p_3 = 3 = c_3$
- $p_1 \neq p_2$, $p_2 \neq p_3$, $p_3 \neq p_1$

All possible correct answers to this test case are: $[1, 2, 3]$, $[1, 3, 2]$, $[2, 1, 3]$, $[2, 3, 1]$, $[3, 1, 2]$, $[3, 2, 1]$.

In the second test case $p = [1, 2, 1, 2]$.

In this sequence $p_1 = a_1$, $p_2 = a_2$, $p_3 = a_3$, $p_4 = a_4$. Also we can see, that no two adjacent elements of the sequence are equal.

In the third test case $p = [1, 3, 4, 3, 2, 4, 2]$.

In this sequence $p_1 = a_1$, $p_2 = a_2$, $p_3 = b_3$, $p_4 = b_4$, $p_5 = b_5$, $p_6 = c_6$, $p_7 = c_7$. Also we can see, that no two adjacent elements of the sequence are equal.

B. Arrays Sum

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

You are given a **non-decreasing** array of **non-negative** integers $a_1, a_2, \ldots, a_n$. Also you are given a positive integer k.

You want to find m **non-decreasing** arrays of **non-negative** integers $b_1, b_2, \ldots, b_m$, such that:

- The size of b_i is equal to n for all $1 \leq i \leq m$.
- For all $1 \leq j \leq n$, $a_j = b_{1,j} + b_{2,j} + \ldots + b_{m,j}$. In the other word, array a is the sum of arrays b_i.
- The number of different elements in the array b_i is at most k for all $1 \leq i \leq m$.

Find the minimum possible value of m, or report that there is no possible m.

Input

The first line contains one integer t ($1 \leq t \leq 100$): the number of test cases.

The first line of each test case contains two integers n, k ($1 \leq n \leq 100$, $1 \leq k \leq n$).

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($0 \leq a_1 \leq a_2 \leq \ldots \leq a_n \leq 100$, $a_n > 0$).

Output

For each test case print a single integer: the minimum possible value of m. If there is no such m, print -1.

Example

input

```
6
4 1
0 0 0 1
3 1
3 3 3
11 3
0 1 2 2 3 3 3 4 4 4 4
5 3
1 2 3 4 5
9 4
2 2 3 5 7 11 13 13 17
10 7
0 1 1 2 3 3 4 5 5 6
```

output

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

Note

In the first test case, there is no possible m, because all elements of all arrays should be equal to 0. But in this case, it is impossible to get $a_4 = 1$ as the sum of zeros.

In the second test case, we can take $b_1 = [3, 3, 3]$. 1 is the smallest possible value of m.

In the third test case, we can take $b_1 = [0, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2]$ and $b_2 = [0, 0, 1, 1, 1, 1, 1, 2, 2, 2, 2]$. It's easy to see, that $a_i = b_{1,i} + b_{2,i}$ for all i and the number of different elements in b_1 and in b_2 is equal to 3 (so it is at most 3). It can be proven that 2 is the smallest possible value of m.

C. Discrete Acceleration

time limit per test: 3 seconds
memory limit per test: 256 megabytes

input: standard input
output: standard output

There is a road with length l meters. The start of the road has coordinate 0, the end of the road has coordinate l.

There are two cars, the first standing at the start of the road and the second standing at the end of the road. They will start driving simultaneously. The first car will drive from the start to the end and the second car will drive from the end to the start.

Initially, they will drive with a speed of 1 meter per second. There are n flags at **different** coordinates $a_1, a_2, \ldots, a_n$. Each time when any of two cars drives through a flag, the speed of that car increases by 1 meter per second.

Find how long will it take for cars to meet (to reach the same coordinate).

Input

The first line contains one integer t ($1 \leq t \leq 10^4$): the number of test cases.

The first line of each test case contains two integers n, l ($1 \leq n \leq 10^5$, $1 \leq l \leq 10^9$): the number of flags and the length of the road.

The second line contains n integers $a_1, a_2, \ldots, a_n$ in the **increasing** order ($1 \leq a_1 < a_2 < \ldots < a_n < l$).

It is guaranteed that the sum of n among all test cases does not exceed 10^5.

Output

For each test case print a single real number: the time required for cars to meet.

Your answer will be considered correct, if its absolute or relative error does not exceed 10^{-6}. More formally, if your answer is a and jury's answer is b, your answer will be considered correct if $\frac{|a-b|}{\max(1,b)} \leq 10^{-6}$.

Example

input

```
5
2 10
1 9
1 10
1
5 7
1 2 3 4 6
2 1000000000
413470354 982876160
9 478
1 10 25 33 239 445 453 468 477
```

output

```
3.000000000000000
3.666666666666667
2.047619047619048
329737645.750000000000000
53.700000000000000
```

Note

In the first test case cars will meet in the coordinate 5.

The first car will be in the coordinate 1 in 1 second and after that its speed will increase by 1 and will be equal to 2 meters per second. After 2 more seconds it will be in the coordinate 5. So, it will be in the coordinate 5 in 3 seconds.

The second car will be in the coordinate 9 in 1 second and after that its speed will increase by 1 and will be equal to 2 meters per second. After 2 more seconds it will be in the coordinate 5. So, it will be in the coordinate 5 in 3 seconds.

In the second test case after 1 second the first car will be in the coordinate 1 and will have the speed equal to 2 meters per second, the second car will be in the coordinate 9 and will have the speed equal to 1 meter per second. So, they will meet after $\frac{9-1}{2+1} = \frac{8}{3}$ seconds. So, the answer is equal to $1 + \frac{8}{3} = \frac{11}{3}$.

D. Searchlights

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

There are n robbers at coordinates (a_1, b_1), (a_2, b_2), ..., (a_n, b_n) and m searchlight at coordinates (c_1, d_1), (c_2, d_2), ..., (c_m, d_m).

In one move you can move each robber to the right (increase a_i of each robber by one) or move each robber up (increase b_i of each robber by one). Note that you should either increase **all** a_i or **all** b_i, you **can't** increase a_i for some points and b_i for some other points.

Searchlight j can see a robber i if $a_i \leq c_j$ and $b_i \leq d_j$.

A configuration of robbers is *safe* if no searchlight can see a robber (i.e. if there is no pair i, j such that searchlight j can see a robber i).

What is the minimum number of moves you need to perform to reach a safe configuration?

Input

The first line of input contains two integers n and m ($1 \le n, m \le 2000$): the number of robbers and the number of searchlight.

Each of the next n lines contains two integers a_i, b_i ($0 \le a_i, b_i \le 10^6$), coordinates of robbers.

Each of the next m lines contains two integers c_i, d_i ($0 \le c_i, d_i \le 10^6$), coordinates of searchlights.

Output

Print one integer: the minimum number of moves you need to perform to reach a safe configuration.

Examples

input

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

output

```
3
```

input

```
2 3
1 6
6 1
10 1
1 10
7 7
```

output

```
4
```

input

```
1 2
0 0
0 0
0 0
```

output

```
1
```

input

```
7 3
0 8
3 8
2 7
0 10
5 5
7 0
3 5
6 6
3 11
11 5
```

output

```
6
```

Note

In the first test, you can move each robber to the right three times. After that there will be one robber in the coordinates $(3, 0)$.

The configuration of the robbers is safe, because the only searchlight can't see the robber, because it is in the coordinates $(2, 3)$ and $3 > 2$.

In the second test, you can move each robber to the right two times and two times up. After that robbers will be in the coordinates $(3, 8)$, $(8, 3)$.

It's easy the see that the configuration of the robbers is safe.

It can be proved that you can't reach a safe configuration using no more than 3 moves.

E. Avoid Rainbow Cycles

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

You are given m sets of integers $A_1, A_2, \ldots, A_m$; elements of these sets are integers between 1 and n, inclusive.

There are two arrays of positive integers $a_1, a_2, \ldots, a_m$ and $b_1, b_2, \ldots, b_n$.

In one operation you can delete an element j from the set A_i and pay $a_i + b_j$ coins for that.

You can make several (maybe none) operations (some sets can become empty).

After that, you will make an edge-colored undirected graph consisting of n vertices. For each set A_i you will add an edge (x, y) with color i for all $x, y \in A_i$ and $x < y$. Some pairs of vertices can be connected with more than one edge, but such edges have different colors.

You call a cycle $i_1 \to e_1 \to i_2 \to e_2 \to \ldots \to i_k \to e_k \to i_1$ (e_j is some edge connecting vertices i_j and i_{j+1} in this graph) *rainbow* if all edges on it have different colors.

Find the minimum number of coins you should pay to get a graph without rainbow cycles.

Input

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

The second line contains m integers $a_1, a_2, \ldots, a_m$ ($1 \le a_i \le 10^9$).

The third line contains n integers $b_1, b_2, \ldots, b_n$ ($1 \le b_i \le 10^9$).

In the each of the next of m lines there are descriptions of sets. In the i-th line the first integer s_i ($1 \le s_i \le n$) is equal to the size of A_i. Then s_i integers follow: the elements of the set A_i. These integers are from 1 to n and distinct.

It is guaranteed that the sum of s_i for all $1 \le i \le m$ does not exceed $2 \cdot 10^5$.

Output

Print one integer: the minimum number of coins you should pay for operations to avoid rainbow cycles in the obtained graph.

Examples

input

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

output

```
11
```

input

```
7 8
3 6 7 9 10 7 239
8 1 9 7 10 2 6 239
3 2 1 3
2 4 1
3 1 3 7
2 4 3
5 3 4 5 6 7
2 5 7
1 8
```

output

```
66
```

Note

In the first test, you can make such operations:

- Delete element 1 from set 1. You should pay $a_1 + b_1 = 5$ coins for that.
- Delete element 1 from set 2. You should pay $a_2 + b_1 = 6$ coins for that.

You pay 11 coins in total. After these operations, the first and the second sets will be equal to $\{2\}$ and the third set will be equal to $\{1, 2\}$.

So, the graph will consist of one edge $(1, 2)$ of color 3.

In the second test, you can make such operations:

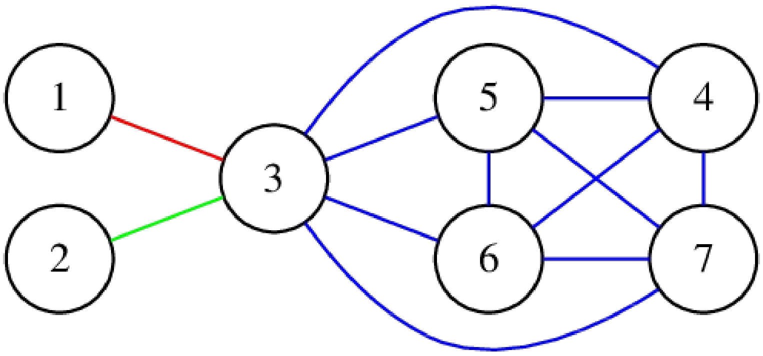
- Delete element 1 from set 1. You should pay $a_1 + b_1 = 11$ coins for that.
- Delete element 4 from set 2. You should pay $a_2 + b_4 = 13$ coins for that.
- Delete element 7 from set 3. You should pay $a_3 + b_7 = 13$ coins for that.
- Delete element 4 from set 4. You should pay $a_4 + b_4 = 16$ coins for that.
- Delete element 7 from set 6. You should pay $a_6 + b_7 = 13$ coins for that.

You pay 66 coins in total.

After these operations, the sets will be:

- $\{2, 3\}$;
- $\{1\}$;
- $\{1, 3\}$;
- $\{3\}$;
- $\{3, 4, 5, 6, 7\}$;
- $\{5\}$;
- $\{8\}$.

We will get the graph:



There are no rainbow cycles in it.

F. Two Different

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

You are given an integer n.

You should find a list of pairs (x_1, y_1), (x_2, y_2), ..., (x_q, y_q) ($1 \leq x_i, y_i \leq n$) satisfying the following condition.

Let's consider some function $f: \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ (we define $\mathbb{N}$ as the set of positive integers). In other words, f is a function that returns a positive integer for a pair of positive integers.

Let's make an array $a_1, a_2, \ldots, a_n$, where $a_i = i$ initially.

You will perform q operations, in i-th of them you will:

1. assign $t = f(a_{x_i}, a_{y_i})$ (t is a temporary variable, it is used **only** for the next two assignments);
2. assign $a_{x_i} = t$;
3. assign $a_{y_i} = t$.

In other words, you need to **simultaneously** change a_{x_i} and a_{y_i} to $f(a_{x_i}, a_{y_i})$. Note that during this process $f(p, q)$ is always the same for a fixed pair of p and q.

In the end, there should be at most two different numbers in the array a.

It should be true for any function f.

Find any possible list of pairs. The number of pairs should not exceed $5 \cdot 10^5$.

Input

The single line contains a single integer n ($1 \leq n \leq 15\,000$).

Output

In the first line print q ($0 \leq q \leq 5 \cdot 10^5$) — the number of pairs.

In each of the next q lines print two integers. In the i-th line print x_i, y_i ($1 \leq x_i, y_i \leq n$).

The condition described in the statement should be satisfied.

If there exists multiple answers you can print any of them.

Examples

input
3

output
1

```
1 2
```

input

```
4
```

output

```
2
1 2
3 4
```

Note

In the first example, after performing the only operation the array a will be $[f(a_1, a_2), f(a_1, a_2), a_3]$. It will always have at most two different numbers.

In the second example, after performing two operations the array a will be $[f(a_1, a_2), f(a_1, a_2), f(a_3, a_4), f(a_3, a_4)]$. It will always have at most two different numbers.

G. Clusterization Counting

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

There are n computers in the company network. They are numbered from 1 to n.

For each pair of two computers $1 \le i < j \le n$ you know the value $a_{i,j}$: the difficulty of sending data between computers i and j. All values $a_{i,j}$ for $i < j$ are different.

You want to separate all computers into k sets $A_1, A_2, \ldots, A_k$, such that the following conditions are satisfied:

- for each computer $1 \le i \le n$ there is **exactly** one set A_j, such that $i \in A_j$;
- for each two pairs of computers (s, f) and (x, y) ($s \ne f$, $x \ne y$), such that s, f, x are from the same set but x and y are from different sets, $a_{s,f} < a_{x,y}$.

For each $1 \le k \le n$ find the number of ways to divide computers into k groups, such that all required conditions are satisfied. These values can be large, so you need to find them by modulo $998\,244\,353$.

Input

The first line contains a single integer n ($1 \le n \le 1500$): the number of computers.

The i-th of the next n lines contains n integers $a_{i,1}, a_{i,2}, \ldots, a_{i,n}$($0 \le a_{i,j} \le \frac{n(n-1)}{2}$).

It is guaranteed that:

- for all $1 \le i \le n$ $a_{i,i} = 0$;
- for all $1 \le i < j \le n$ $a_{i,j} > 0$;
- for all $1 \le i < j \le n$ $a_{i,j} = a_{j,i}$;
- all $a_{i,j}$ for $i < j$ are different.

Output

Print n integers: the k-th of them should be equal to the number of possible ways to divide computers into k groups, such that all required conditions are satisfied, modulo $998\,244\,353$.

Examples

input

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

output

```
1 0 1 1
```

input

```
7
0 1 18 15 19 12 21
1 0 16 13 17 20 14
18 16 0 2 7 10 9
15 13 2 0 6 8 11
19 17 7 6 0 4 5
12 20 10 8 4 0 3
21 14 9 11 5 3 0
```

output

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

Note

Here are all possible ways to separate all computers into 4 groups in the second example:

- $\{1, 2\}, \{3, 4\}, \{5\}, \{6, 7\}$;
- $\{1\}, \{2\}, \{3, 4\}, \{5, 6, 7\}$;
- $\{1, 2\}, \{3\}, \{4\}, \{5, 6, 7\}$.

H. Rainbow Triples

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

You are given a sequence $a_1, a_2, \ldots, a_n$ of non-negative integers.

You need to find the largest number m of triples (i_1, j_1, k_1), (i_2, j_2, k_2), ..., (i_m, j_m, k_m) such that:

- $1 \le i_p < j_p < k_p \le n$ for each p in $1, 2, \ldots, m$;
- $a_{i_p} = a_{k_p} = 0$, $a_{j_p} \ne 0$;
- all $a_{j_1}, a_{j_2}, \ldots, a_{j_m}$ are different;
- all $i_1, j_1, k_1, i_2, j_2, k_2, \ldots, i_m, j_m, k_m$ are different.

Input

The first line of input contains one integer t ($1 \le t \le 500\,000$): the number of test cases.

The first line of each test case contains one integer n ($1 \le n \le 500\,000$).

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le n$).

The total sum of n is at most $500\,000$.

Output

For each test case, print one integer m: the largest number of proper triples that you can find.

Example

input

```
8
1
1
2
0 0
3
0 1 0
6
0 0 1 2 0 0
6
0 1 0 0 1 0
6
0 1 3 2 0 0
6
0 0 0 0 5 0
12
0 1 0 2 2 2 0 0 3 3 4 0
```

output

```
0
0
1
2
1
1
1
2
```

Note

In the first two test cases, there are not enough elements even for a single triple, so the answer is 0.

In the third test case we can select one triple $(1, 2, 3)$.

In the fourth test case we can select two triples $(1, 3, 5)$ and $(2, 4, 6)$.

In the fifth test case we can select one triple $(1, 2, 3)$. We can't select two triples $(1, 2, 3)$ and $(4, 5, 6)$, because $a_2 = a_5$.

I. Bitwise Magic

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

You are given a positive integer k and an array $a_1, a_2, \ldots, a_n$ of non-negative distinct integers not smaller than k and not greater than $2^c - 1$.

In each of the next k seconds, one element is chosen randomly equiprobably out of all n elements and decreased by 1.

For each integer x, $0 \leq x \leq 2^c - 1$, you need to find the probability that in the end the bitwise XOR of all elements of the array is equal to x.

Each of these values can be represented as an irreducible fraction $\frac{p}{q}$, and you need to find the value of $p \cdot q^{-1}$ modulo $998\,244\,353$.

Input

The first line of input contains three integers n, k, c ($1 \leq n \leq (2^c - k)$, $1 \leq k \leq 16$, $1 \leq c \leq 16$).

The second line contains n distinct integers $a_1, a_2, \ldots, a_n$ ($k \leq a_i \leq 2^c - 1$).

Output

Print 2^c integers: the probability that the bitwise XOR is equal to x in the end for x in $\{0, 1, \ldots, 2^c - 1\}$ modulo $998\,244\,353$.

Example

input

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

output

```
0 0 0 748683265 0 499122177 0 748683265
```