A. Fibonacciness

```
Input file:     standard input
Output file:    standard output
Time limit:     1 second
Memory limit:   256 megabytes
```

There is an array of 5 integers. Initially, you only know a_1, a_2, a_4, a_5. You may set a_3 to any positive integer, negative integer, or zero. The *Fibonacciness* of the array is the number of integers i ($1 \leq i \leq 3$) such that $a_{i+2} = a_i + a_{i+1}$. Find the maximum *Fibonacciness* over all integer values of a_3.

Input

The first line contains an integer t ($1 \leq t \leq 500$) — the number of test cases.

The only line of each test case contains four integers a_1, a_2, a_4, a_5 ($1 \leq a_i \leq 100$).

Output

For each test case, output the maximum *Fibonacciness* on a new line.

Standard Input	Standard Output
6 1 1 3 5 1 3 2 1 8 10 28 100 100 1 100 1 1 100 1 100 100 100 100 100	3 2 2 1 1 2

Note

In the first test case, we can set a_3 to 2 to achieve the maximal *Fibonacciness* of 3.

In the third test case, it can be shown that 2 is the maximum *Fibonacciness* that can be achieved. This can be done by setting a_3 to 18.

,

B. Farmer John's Card Game

```
Input file:     standard input
Output file:    standard output
Time limit:     2 seconds
Memory limit:   256 megabytes
```

Farmer John's n cows are playing a card game! Farmer John has a deck of $n \cdot m$ cards numbered from 0 to $n \cdot m - 1$. He distributes m cards to each of his n cows.

Farmer John wants the game to be fair, so each cow should only be able to play 1 card per round. He decides to determine a *turn order*, determined by a permutation* p of length n, such that the p_i'th cow will be the i'th cow to place a card on top of the center pile in a round.

In other words, the following events happen in order in each round:

- The p_1'th cow places any card from their deck on top of the center pile.
- The p_2'th cow places any card from their deck on top of the center pile.
- ...
- The p_n'th cow places any card from their deck on top of the center pile.

There is a catch. Initially, the center pile contains a card numbered -1. In order to place a card, the number of the card must be greater than the number of the card on top of the center pile. Then, the newly placed card becomes the top card of the center pile. If a cow cannot place any card in their deck, the game is considered to be lost.

Farmer John wonders: does there exist p such that it is possible for all of his cows to empty their deck after playing all m rounds of the game? If so, output any valid p. Otherwise, output -1.

---

* A permutation of length n contains each integer from 1 to n exactly once

Input

The first line contains an integer t ($1 \leq t \leq 400$) — the number of test cases.

The first line of each test case contains two integers n and m ($1 \leq n \cdot m \leq 2\,000$) — the number of cows and the number of cards each cow receives.

The following n lines contain m integers each – the cards received by each cow. It is guaranteed all given numbers (across all n lines) are distinct and in the range from 0 to $n \cdot m - 1$, inclusive.

It is guaranteed the sum of $n \cdot m$ over all test cases does not exceed $2\,000$.

Output

For each test case, output the following on a new line:

- If p exists, output n space-separated integers $p_1, p_2, \ldots, p_n$.
- Otherwise, output -1.

Standard Input	Standard Output

4 2 3 0 4 2 1 5 3 1 1 0 2 2 1 2 0 3 4 1 1 2 0 3	1 2 1 -1 3 1 2 4

Note

In the first test case, one turn order that allows all cards to be played is by having the first cow go before the second cow. The cards played will be $0 \to 1 \to 2 \to 3 \to 4 \to 5$.

In the second test case, there is only one cow, so having the cow play all of her cards in increasing order will empty the deck.

In the third test case, it can be shown there is no valid turn order that allows all cards to be played.

,

C. Game of Mathletes

Input file: standard input
Output file: standard output
Time limit: 2 seconds
Memory limit: 256 megabytes

Alice and Bob are playing a game. There are n (n **is even**) integers written on a blackboard, represented by $x_1, x_2, \ldots, x_n$. There is also a given integer k and an integer *score* that is initially 0. The game lasts for $\frac{n}{2}$ turns, in which the following events happen sequentially:

- Alice selects an integer from the blackboard and erases it. Let's call Alice's chosen integer a.
- Bob selects an integer from the blackboard and erases it. Let's call Bob's chosen integer b.
- If $a + b = k$, add 1 to *score*.

Alice is playing to minimize the *score* while Bob is playing to maximize the *score*. Assuming both players use optimal strategies, what is the *score* after the game ends?

Input

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

The first line of each test case contains two integers n and k ($2 \leq n \leq 2 \cdot 10^5, 1 \leq k \leq 2 \cdot n$, n is even).

The second line of each test case contains n integers $x_1, x_2, \ldots, x_n$ ($1 \leq x_i \leq n$) — the integers on the blackboard.

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 *score* if both players play optimally.

Standard Input	Standard Output
4 4 4 1 2 3 2 8 15 1 2 3 4 5 6 7 8 6 1 1 1 1 1 1 1 16 9 3 1 4 1 5 9 2 6 5 3 5 8 9 7 9 3	2 1 0 4

Note

In the first test case, one way the game may go is as follows:

- Alice selects 1 and Bob selects 3. The score increases as $1 + 3 = 4$. Now the two integers remaining on the blackboard are 2 and 2.
- Alice and Bob both select 2. The score increases as $2 + 2 = 4$.
- The game ends as the blackboard now has no integers.

In the third test case, it is impossible for the sum of Alice and Bob's selected integers to be 1, so we answer 0.

Note that this is just an example of how the game may proceed for demonstration purposes. This may not be Alice or Bob's most optimal strategies.

,

D. Subtract Min Sort

Input file: standard input
Output file: standard output
Time limit: 2 seconds
Memory limit: 256 megabytes

You are given a sequence a consisting of n positive integers.

You can perform the following operation any number of times.

- Select an index i ($1 \leq i < n$), and subtract $\min(a_i, a_{i+1})$ from both a_i and a_{i+1}.

Determine if it is possible to make the sequence **non-decreasing** by using the operation any number of times.

Input

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

The first line of each test case contains a single integer n ($2 \leq n \leq 2 \cdot 10^5$).

The second line of each test case contains $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 10^9$).

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

Output

If it is possible to make the sequence **non-decreasing**, print "YES" on a new line. Otherwise, print "NO" on a new line.

You can output the answer in any case. For example, the strings "yEs", "yes", and "Yes" will also be recognized as positive responses.

Standard Input	Standard Output
5 5 1 2 3 4 5 4 4 3 2 1 4 4 5 2 3 8 4 5 4 5 4 5 4 5 9 9 9 8 2 4 4 3 5 3	YES NO YES YES NO

Note

In the first test case, the array is already sorted.

In the second test case, we can show that it is impossible.

In the third test case, after performing an operation on $i = 1$, the array becomes $[0, 1, 2, 3]$, which is now in nondecreasing order.

,

E. Graph Composition

Input file: standard input
Output file: standard output
Time limit: 2 seconds
Memory limit: 256 megabytes

You are given two simple undirected graphs F and G with n vertices. F has m_1 edges while G has m_2 edges. You may perform one of the following two types of operations any number of times:

- Select two integers u and v ($1 \leq u, v \leq n$) such that there is an edge between u and v in F. Then, remove that edge from F.
- Select two integers u and v ($1 \leq u, v \leq n$) such that there is no edge between u and v in F. Then, add an edge between u and v in F.

Determine the minimum number of operations required such that for all integers u and v ($1 \leq u, v \leq n$), there is a path from u to v in F **if and only if** there is a path from u to v in G.

Input

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

The first line of each test case contains three integers n, m_1, and m_2 ($1 \leq n \leq 2 \cdot 10^5$, $0 \leq m_1, m_2 \leq 2 \cdot 10^5$) — the number of vertices, the number of edges in F, and the number of edges in G.

The following m_1 lines each contain two integers u and v ($1 \leq u, v \leq n$) — there is an edge between u and v in F. It is guaranteed that there are no repeated edges or self loops.

The following m_2 lines each contain two integers u and v ($1 \leq u, v \leq n$) — there is an edge between u and v in G. It is guaranteed that there are no repeated edges or self loops.

It is guaranteed that the sum of n, the sum of m_1, and the sum of m_2 over all test cases do not exceed $2 \cdot 10^5$.

Output

For each test case, output a single integer denoting the minimum operations required on a new line.

Standard Input	Standard Output
5 3 2 1 1 2 2 3 1 3 2 1 1 1 2 1 2 3 2 0 3 2 1 2 1 0 0 3 3 1 1 2	3 0 2 0 2

1 3 2 3 1 2	

Note

In the first test case you can perform the following three operations:

1. Add an edge between vertex 1 and vertex 3.
2. Remove the edge between vertex 1 and vertex 2.
3. Remove the edge between vertex 2 and vertex 3.

It can be shown that fewer operations cannot be achieved.
In the second test case, F and G already fulfill the condition in the beginning.

In the fifth test case, the edges from 1 to 3 and from 2 to 3 must both be removed.

,

F. Multiplicative Arrays

Input file: standard input
Output file: standard output
Time limit: 4 seconds
Memory limit: 512 megabytes

You're given integers k and n. For each integer x from 1 to k, count the number of integer arrays a such that all of the following are satisfied:

- $1 \le |a| \le n$ where $|a|$ represents the length of a.
- $1 \le a_i \le k$ for all $1 \le i \le |a|$.
- $a_1 \times a_2 \times \cdots \times a_{|a|} = x$ (i.e., the product of all elements is x).

Note that two arrays b and c are different if either their lengths are different, or if there exists an index $1 \le i \le |b|$ such that $b_i \neq c_i$.

Output the answer modulo $998\,244\,353$.

Input

The first line contains an integer t ($1 \le t \le 10^3$) — the number of independent test cases.

The only line of each test case contains two integers k and n ($1 \le k \le 10^5, 1 \le n \le 9 \cdot 10^8$).

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

Output

For each test case, output k space-separated integers on a new line: the number of arrays for $x = 1, 2, \ldots, k$, modulo $998\,244\,353$.

Standard Input	Standard Output
3 2 2 4 3 10 6969420	2 3 3 6 6 10 6969420 124188773 124188773 729965558 124188773 337497990 124188773 50981194 729965558 337497990

Note

In the first test case, there are 2 arrays a with $|a| \le 2$ and the product of elements equal to 1:

- $[1]$
- $[1, 1]$

There are 3 arrays a with $|a| \le 2$ and the product of elements equal to 2:

- $[2]$
- $[1, 2]$
- $[2, 1]$

,

G. Bugged Sort

Input file: standard input
Output file: standard output
Time limit: 4 seconds
Memory limit: 1024 megabytes

Today, Alice has given Bob arrays for him to sort in increasing order again! At this point, no one really knows how many times she has done this.

Bob is given two sequences a and b, both of length n. All integers in the range from 1 to $2n$ appear exactly once in either a or b. In other words, the concatenated* sequence $a + b$ is a permutation† of length $2n$.

Bob must sort **both sequences** in increasing order **at the same time** using Alice's `swap` function. Alice's `swap` function is implemented as follows:

- Given two indices i and j ($i \neq j$), it swaps a_i with b_j, and swaps b_i with a_j.

Given sequences a and b, please determine if both sequences can be sorted in increasing order simultaneously after using Alice's `swap` function any number of times.

---

*The concatenated sequence $a + b$ denotes the sequence $[a_1, a_2, a_3, \ldots, b_1, b_2, b_3, \ldots]$.

†A permutation of length m contains all integers from 1 to m in some order.

Input

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

The first line of each test case contains a single integer n ($3 \leq n \leq 2 \cdot 10^5$).

The second line of each test case contains $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 2n$).

The third line of each test case contains $b_1, b_2, \ldots, b_n$ ($1 \leq b_i \leq 2n$).

It is guaranteed that all integers in the range $[1, 2n]$ appear exactly once in either a or b.

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

Output

If it is possible to sort both sequences simultaneously, print "YES" on a new line. Otherwise, print "NO" on a new line.

You can output the answer in any case. For example, the strings "yEs", "yes", and "Yes" will also be recognized as positive responses.

Standard Input	Standard Output
5 3 2 1 3 4 6 5 3 2 1 5 4 3 6	NO YES NO YES YES

4 1 6 4 3 5 2 8 7 4 5 3 7 1 8 6 4 2 7 5 1 9 12 3 13 7 2 4 11 14 6 10 8	

Note

In the first test case, it can be shown that it is impossible.

In the second test case, Bob can perform one operation with indices $i = 1$ and $j = 2$. The arrays become $[3, 4, 5]$ and $[1, 2, 6]$ respectively. Both arrays are now sorted.