سوالات

مسابقه درون دانشگاهی

ICPC

دی ماه 1403

Problem A. Division?

Time limit 1000 ms
Mem limit 262144 kB

Codeforces separates its users into 4 divisions by their rating:

- For Division 1: $1900 \leq \text{rating}$
- For Division 2: $1600 \leq \text{rating} \leq 1899$
- For Division 3: $1400 \leq \text{rating} \leq 1599$
- For Division 4: $\text{rating} \leq 1399$

Given a rating, print in which division the rating belongs.

Input

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

The description of each test consists of one line containing one integer rating ($-5000 \leq \text{rating} \leq 5000$).

Output

For each test case, output a single line containing the correct division in the format "`Division X`", where X is an integer between 1 and 4 representing the division for the corresponding rating.

Examples

Input	Output
7 -789 1299 1300 1399 1400 1679 2300	Division 4 Division 4 Division 4 Division 4 Division 3 Division 2 Division 1

Note

For test cases $1 - 4$, the corresponding ratings are $-789, 1299, 1300, 1399$, so all of them are in division 4.

For the fifth test case, the corresponding rating is 1400, so it is in division 3.

For the sixth test case, the corresponding rating is 1679, so it is in division 2.

For the seventh test case, the corresponding rating is 2300, so it is in division 1.

Problem B. Triple

Time limit 1000 ms
Mem limit 262144 kB

Given an array a of n elements, print any value that appears at least three times or print `-1` if there is no such value.

Input

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

The first line of each test case contains an integer n ($1 \le n \le 2 \cdot 10^5$) — the length of the array.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$) — the elements of the array.

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

Output

For each test case, print any value that appears at least three times or print `-1` if there is no such value.

Examples

Input	Output
7 1 1 3 2 2 2 7 2 2 3 3 4 2 2 8 1 4 3 4 3 2 4 1 9 1 1 1 2 2 2 3 3 3 5 1 5 2 4 3 4 4 4 4 4	-1 2 2 4 3 -1 4

Note

In the first test case there is just a single element, so it can't occur at least three times and the answer is `-1`.

In the second test case, all three elements of the array are equal to 2, so 2 occurs three times, and so the answer is 2.

For the third test case, 2 occurs four times, so the answer is 2.

For the fourth test case, 4 occurs three times, so the answer is 4.

For the fifth test case, 1, 2 and 3 all occur at least three times, so they are all valid outputs.

For the sixth test case, all elements are distinct, so none of them occurs at least three times and the answer is `-1`.

Problem C. Sum of Round Numbers

Time limit 1000 ms
Mem limit 262144 kB

A positive (strictly greater than zero) integer is called *round* if it is of the form `d00...0`. In other words, a positive integer is round if all its digits except the leftmost (most significant) are equal to zero. In particular, all numbers from 1 to 9 (inclusive) are round.

For example, the following numbers are round: 4000, 1, 9, 800, 90. The following numbers are **not** round: 110, 707, 222, 1001.

You are given a positive integer n ($1 \le n \le 10^4$). Represent the number n as a sum of round numbers using the minimum number of summands (addends). In other words, you need to represent the given number n as a sum of the least number of terms, each of which is a round number.

Input

The first line contains an integer t ($1 \le t \le 10^4$) — the number of test cases in the input. Then t test cases follow.

Each test case is a line containing an integer n ($1 \le n \le 10^4$).

Output

Print t answers to the test cases. Each answer must begin with an integer k — the minimum number of summands. Next, k terms must follow, each of which is a round number, and their sum is n. The terms can be printed in any order. If there are several answers, print any of them.

Examples

Input	Output
5 5009 7 9876 10000 10	2 5000 9 1 7 4 800 70 6 9000 1 10000 1 10

Problem D. Plus One on the Subset

Time limit 2000 ms
Mem limit 262144 kB

Polycarp got an array of integers $a[1 \dots n]$ as a gift. Now he wants to perform a certain number of operations (possibly zero) so that all elements of the array become the same (that is, to become $a_1 = a_2 = \dots = a_n$).

- In one operation, he can take some indices in the array and increase the elements of the array at those indices by 1.

For example, let $a = [4, 2, 1, 6, 2]$. He can perform the following operation: select indices 1, 2, and 4 and increase elements of the array in those indices by 1. As a result, in one operation, he can get a new state of the array $a = [5, 3, 1, 7, 2]$.

What is the minimum number of operations it can take so that all elements of the array become equal to each other (that is, to become $a_1 = a_2 = \dots = a_n$)?

Input

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

The following are descriptions of the input test cases.

The first line of the description of each test case contains one integer n ($1 \le n \le 50$) — the array a.

The second line of the description of each test case contains n integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le 10^9$) — elements of the array a.

Output

For each test case, print one integer — the minimum number of operations to make all elements of the array a equal.

Examples

Input	Output
3 6 3 4 2 4 1 2 3 1000 1002 998 2 12 11	3 4 1

Note

First test case:

- $a = [3, 4, 2, 4, 1, 2]$ take a_3, a_5 and perform an operation plus one on them, as a result we get $a = [3, 4, 3, 4, 2, 2]$.
- $a = [3, 4, 3, 4, 2, 2]$ we take a_1, a_5, a_6 and perform an operation on them plus one, as a result we get $a = [4, 4, 3, 4, 3, 3]$.
- $a = [4, 4, 3, 4, 3, 3]$ we take a_3, a_5, a_6 and perform an operation on them plus one, as a result we get $a = [4, 4, 4, 4, 4, 4]$.

There are other sequences of 3 operations, after the application of which all elements become equal.

Second test case:

- $a = [1000, 1002, 998]$ 2 times we take a_1, a_3 and perform an operation plus one on them, as a result we get $a = [1002, 1002, 1000]$.
- $a = [1002, 1002, 1000]$ also take a_3 2 times and perform an operation plus one on it, as a result we get $a = [1002, 1002, 1002]$.

Third test case:

- $a = [12, 11]$ take a_2 and perform an operation plus one on it, as a result we get $a = [12, 12]$.

Problem E. Transfusion

Time limit 2000 ms
Mem limit 262144 kB

You are given an array a of length n. In one operation, you can pick an index i from 2 to $n - 1$ inclusive, and do one of the following actions:

- Decrease a_{i-1} by 1, then increase a_{i+1} by 1.
- Decrease a_{i+1} by 1, then increase a_{i-1} by 1.

After each operation, all the values must be non-negative. Can you make all the elements equal after any number of operations?

Input

First line of input consists of one integer t ($1 \leq t \leq 10^4$) — the number of test cases.

First line of each test case consists of one integer n ($3 \leq n \leq 2 \cdot 10^5$).

Second line of each test case consists of n integers a_i ($1 \leq a_i \leq 10^9$).

It is guaranteed that the sum of n of all test cases doesn't exceed $2 \cdot 10^5$.

Output

For each test case, print "`YES`" without quotation marks if it is possible to make all the elements equal after any number of operations; otherwise, print "`NO`" without quotation marks.

You can print answers in any register: "`yes`", "`YeS`", "`nO`" — will also be considered correct.

Examples

Input	Output
8 3 3 2 1 3 1 1 3 4 1 2 5 4 4 1 6 6 1 5 6 2 1 4 2 4 1 4 2 1 5 3 1 2 1 3 3 2 4 2	YES NO YES NO YES NO NO NO

Problem F. Progressive Square

Time limit 2000 ms
Mem limit 262144 kB

A *progressive square* of size n is an $n \times n$ matrix. Maxim chooses three integers $a_{1,1}$, c, and d and constructs a *progressive square* according to the following rules:

$$a_{i+1,j} = a_{i,j} + c$$

$$a_{i,j+1} = a_{i,j} + d$$

For example, if $n = 3$, $a_{1,1} = 1$, $c = 2$, and $d = 3$, then the *progressive square* looks as follows:

$$\begin{pmatrix} 1 & 4 & 7 \\ 3 & 6 & 9 \\ 5 & 8 & 11 \end{pmatrix}$$

Last month Maxim constructed a *progressive square* and remembered the values of n, c, and d. Recently, he found an array b of n^2 integers in random order and wants to make sure that these elements are the elements of **that specific** square.

It can be shown that for any values of n, $a_{1,1}$, c, and d, there exists exactly one *progressive square* that satisfies all the rules.

Input

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

The first line of each test case contains three integers n, c, and d ($2 \le n \le 500$, $1 \le c, d \le 10^6$) — the size of the square and the values of c and d as described in the statement.

The second line of each test case contains $n \cdot n$ integers $b_1, b_2, \ldots, b_{n \cdot n}$ ($1 \le b_i \le 10^9$) — the elements found by Maxim.

It is guaranteed that the sum of n^2 over all test cases does not exceed $25 \cdot 10^4$.

Output

For each test case, output "`YES`" in a separate line if a *progressive square* for the given n, c, and d can be constructed from the array elements a, otherwise output "`NO`".

You can output each letter in any case (lowercase or uppercase). For example, the strings "yEs", "yes", "Yes", and "YES" will be accepted as a positive answer.

Examples

Input	Output
5 3 2 3 3 9 6 5 7 1 10 4 8 3 2 3 3 9 6 5 7 1 11 4 8 2 100 100 400 300 400 500 3 2 3 3 9 6 6 5 1 11 4 8 4 4 4 15 27 7 19 23 23 11 15 7 3 19 23 11 15 11 15	NO YES YES NO NO

Problem G. Tree Cutting

Time limit 3000 ms
Mem limit 524288 kB

You are given a tree with n vertices.

Your task is to find the maximum number x such that it is possible to remove exactly k edges from this tree in such a way that the size of each remaining connected component[†] is at least x.

[†] Two vertices v and u are in the same connected component if there exists a sequence of numbers $t_1, t_2, \ldots, t_k$ of arbitrary length k, such that $t_1 = v$, $t_k = u$, and for each i from 1 to $k - 1$, vertices t_i and t_{i+1} are connected by an edge.

Input

Each test consists of several sets of input data. The first line contains a single integer t ($1 \leq t \leq 10^4$) — the number of sets of input data. This is followed by a description of the sets of input data.

The first line of each set of input data contains two integers n and k ($1 \leq k < n \leq 10^5$) — the number of vertices in the tree and the number of edges to be removed.

Each of the next $n - 1$ lines of each set of input data contains two integers v and u ($1 \leq v, u \leq n$) — the next edge of the tree.

It is guaranteed that the sum of the values of n for all sets of input data does not exceed 10^5.

Output

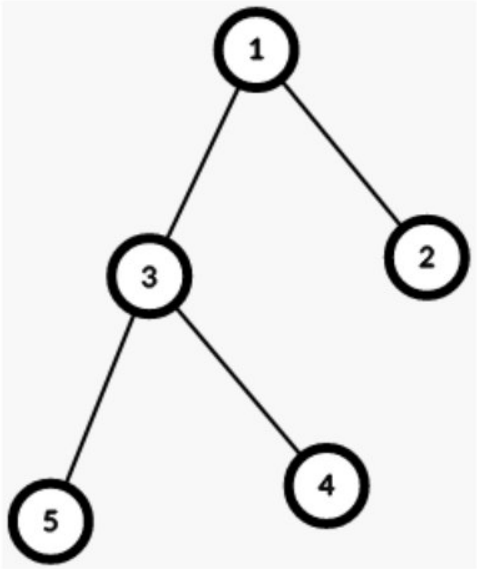
For each set of input data, output a single line containing the maximum number x such that it is possible to remove exactly k edges from the tree in such a way that the size of each remaining connected component is at least x.

Examples

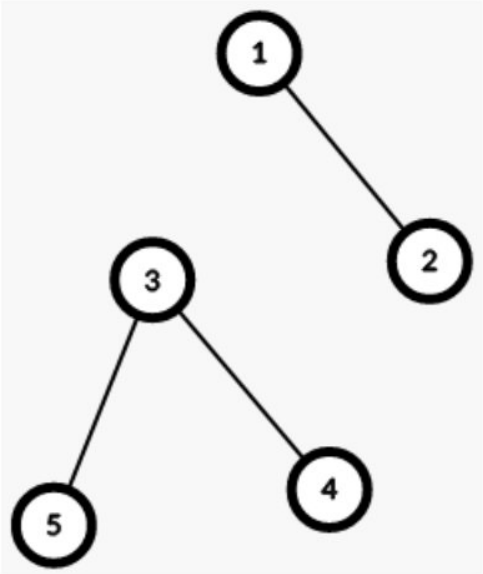
Input	Output
6 5 1 1 2 1 3 3 4 3 5 2 1 1 2 6 1 1 2 2 3 3 4 4 5 5 6 3 1 1 2 1 3 8 2 1 2 1 3 2 4 2 5 3 6 3 7 3 8 6 2 1 2 2 3 1 4 4 5 5 6	2 1 3 1 1 2

Note

The tree in the first set of input data:



After removing the edge $1 — 3$, the tree will look as follows:



The tree has split into two connected components. The first component consists of two vertices: 1 and 2. The second connected component consists of three vertices: 3, 4 and 5. In both connected components, there are at least two vertices. It can be shown that the answer 3 is not achievable, so the answer is 2.

Problem H. Counting Factorizations

Time limit 4000 ms
Mem limit 262144 kB

The prime factorization of a positive integer m is the unique way to write it as $m = p_1^{e_1} \cdot p_2^{e_2} \cdot \ldots \cdot p_k^{e_k}$, where $p_1, p_2, \ldots, p_k$ are prime numbers, $p_1 < p_2 < \ldots < p_k$ and $e_1, e_2, \ldots, e_k$ are positive integers.

For each positive integer m, $f(m)$ is defined as the multiset of all numbers in its prime factorization, that is $f(m) = \{p_1, e_1, p_2, e_2, \ldots, p_k, e_k\}$.

For example, $f(24) = \{2, 3, 3, 1\}$, $f(5) = \{1, 5\}$ and $f(1) = \{\}$.

You are given a list consisting of $2n$ integers $a_1, a_2, \ldots, a_{2n}$. Count how many positive integers m satisfy that $f(m) = \{a_1, a_2, \ldots, a_{2n}\}$. Since this value may be large, print it modulo $998\,244\,353$.

Input

The first line contains one integer n ($1 \le n \le 2022$).

The second line contains $2n$ integers $a_1, a_2, \ldots, a_{2n}$ ($1 \le a_i \le 10^6$) — the given list.

Output

Print one integer, the number of positive integers m satisfying $f(m) = \{a_1, a_2, \ldots, a_{2n}\}$ modulo $998\,244\,353$.

Examples

Input	Output
2 1 3 2 3	2

Input	Output
2 2 2 3 5	5

Input	Output
1 1 4	0

Note

In the first sample, the two values of m such that $f(m) = \{1, 2, 3, 3\}$ are $m = 24$ and $m = 54$. Their prime factorizations are $24 = 2^3 \cdot 3^1$ and $54 = 2^1 \cdot 3^3$.

In the second sample, the five values of m such that $f(m) = \{2, 2, 3, 5\}$ are $200, 225, 288, 500$ and 972.

In the third sample, there is no value of m such that $f(m) = \{1, 4\}$. Neither 1^4 nor 4^1 are prime factorizations because 1 and 4 are **not** primes.