5

A. Medium Number

1 second, 256 megabytes

Given three **distinct** integers a, b, and c, find the medium number between all of them.

The medium number is the number that is neither the minimum nor the maximum of the given three numbers.

For example, the median of $5, 2, 6$ is 5, since the minimum is 2 and the maximum is 6.

Input
The first line contains a single integer t ($1 \le t \le 6840$) — the number of test cases.

The description of each test case consists of three **distinct** integers a, b, c ($1 \le a, b, c \le 20$).

Output
For each test case, output a single integer — the medium number of the three numbers.

input

```
9
5 2 6
14 3 4
20 2 1
1 2 3
11 19 12
10 8 20
6 20 3
4 1 3
19 8 4
```

output

```
5
4
2
2
12
10
6
3
8
```

B. Atilla's Favorite Problem

1 second, 256 megabytes

In order to write a string, Atilla needs to first learn all letters that are contained in the string.

Atilla needs to write a message which can be represented as a string s. He asks you what is the minimum alphabet size required so that one can write this message.

The alphabet of size x ($1 \le x \le 26$) contains **only the first** x Latin letters. For example an alphabet of size 4 contains **only** the characters `a`, `b`, `c` and `d`.

Input
The first line contains a single integer t ($1 \le t \le 1000$) — the number of test cases.

The first line of each test case contains a single integer n ($1 \le n \le 100$) — the length of the string.

The second line of each test case contains a string s of length n, consisting of lowercase Latin letters.

Output
For each test case, output a single integer — the minimum alphabet size required so that Atilla can write his message s.

input

```
5
1
a
4
down
10
codeforces
3
bcf
5
zzzzz
```

output

```
1
23
19
6
26
```

For the first test case, Atilla needs to know only the character `a`, so the alphabet of size 1 which only contains `a` is enough.

For the second test case, Atilla needs to know the characters `d`, `o`, `w`, `n`. The smallest alphabet size that contains all of them is 23 (such alphabet can be represented as the string `abcdefghijklmnopqrstuvw`).

C. Advantage

2 seconds, 256 megabytes

There are n participants in a competition, participant i having a strength of s_i.

Every participant wonders how much of an advantage they have over the other best participant. In other words, each participant i wants to know the difference between s_i and s_j, where j is the strongest participant in the competition, not counting i (a difference can be negative).

So, they ask you for your help! For each i ($1 \le i \le n$) output the difference between s_i and the maximum strength of any participant other than participant i.

Input
The input consists of multiple test cases. The first line contains an integer t ($1 \le t \le 1000$) — the number of test cases. The descriptions of the test cases follow.

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

The following line contains n space-separated positive integers s_1, s_2, ..., s_n ($1 \le s_i \le 10^9$) — the strengths of the participants.

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 n space-separated integers. For each i ($1 \le i \le n$) output the difference between s_i and the maximum strength of any other participant.

input

```
5
4
4 7 3 5
2
1 2
5
1 2 3 4 5
3
4 9 4
4
4 4 4 4
```

output

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

For the first test case:

- The first participant has a strength of 4 and the largest strength of a participant different from the first one is 7, so the answer for the first participant is $4 - 7 = -3$.
- The second participant has a strength of 7 and the largest strength of a participant different from the second one is 5, so the answer for the second participant is $7 - 5 = 2$.
- The third participant has a strength of 3 and the largest strength of a participant different from the third one is 7, so the answer for the third participant is $3 - 7 = -4$.
- The fourth participant has a strength of 5 and the largest strength of a participant different from the fourth one is 7, so the answer for the fourth participant is $5 - 7 = -2$.

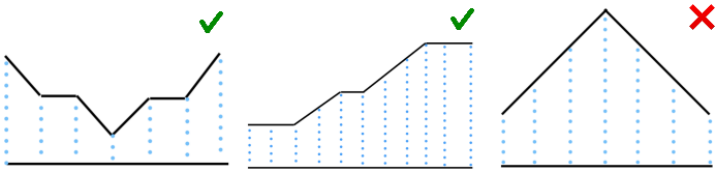
D. Challenging Valleys

2 seconds, 256 megabytes

You are given an array $a[0 \ldots n-1]$ of n integers. This array is called a "*valley*" if there exists **exactly one** subarray $a[l \ldots r]$ such that:

- $0 \le l \le r \le n-1$,
- $a_l = a_{l+1} = a_{l+2} = \cdots = a_r$,
- $l = 0$ or $a_{l-1} > a_l$,
- $r = n-1$ or $a_r < a_{r+1}$.

Here are three examples:



The first image shows the array $[3, 2, 2, 1, 2, 2, 3]$, it **is a valley** because only subarray with indices $l = r = 3$ satisfies the condition.

The second image shows the array $[1, 1, 1, 2, 3, 3, 4, 5, 6, 6, 6]$, it **is a valley** because only subarray with indices $l = 0, r = 2$ satisfies the codition.

The third image shows the array $[1, 2, 3, 4, 3, 2, 1]$, it **is not a valley** because two subarrays $l = r = 0$ and $l = r = 6$ that satisfy the condition.

You are asked whether the given array is a valley or not.

Note that we consider the array to be indexed from 0.

Input

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

The first line of each test case contains a single 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_i ($1 \le a_i \le 10^9$) — the elements of the array.

It is guaranteed that the sum of n over all test cases is smaller than $2 \cdot 10^5$.

Output

For each test case, output "`YES`" (without quotes) if the array is a valley, and "`NO`" (without quotes) otherwise.

You can output the answer in any case (for example, the strings "`yEs`", "`yes`", "`Yes`" and "`YES`" will be recognized as a positive answer).

input

```
6
7
3 2 2 1 2 2 3
11
1 1 1 2 3 3 4 5 6 6 6
7
1 2 3 4 3 2 1
7
9 7 4 6 9 9 10
1
1000000000
8
9 4 4 5 9 4 9 10
```

output

```
YES
YES
NO
YES
YES
NO
```

The first three test cases are explained in the statement.

E. Binary Inversions

2 seconds, 256 megabytes

You are given a binary array[†] of length n. You are allowed to perform one operation on it **at most once**. In an operation, you can choose any element and flip it: turn a 0 into a 1 or vice-versa.

What is the maximum number of inversions[‡] the array can have after performing **at most one** operation?

[†] A binary array is an array that contains only zeroes and ones.

[‡] The number of inversions in an array is the number of pairs of indices i, j such that $i < j$ and $a_i > a_j$.

Input

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

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 following line contains n space-separated positive integers a_1, a_2,..., a_n ($0 \le a_i \le 1$) — 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, output a single integer — the maximum number of inversions the array can have after performing **at most one** operation.

input
5 4 1 0 1 0 6 0 1 0 0 1 0 2 0 0 8 1 0 1 1 0 0 0 1 3 1 1 1

output
3 7 1 13 2

For the first test case, the inversions are initially formed by the pairs of indices $(1, 2)$, $(1, 4)$, $(3, 4)$, being a total of 3, which already is the maximum possible.

For the second test case, the inversions are initially formed by the pairs of indices $(2, 3)$, $(2, 4)$, $(2, 6)$, $(5, 6)$, being a total of four. But, by flipping the first element, the array becomes $1, 1, 0, 0, 1, 0$, which has the inversions formed by the pairs of indices $(1, 3)$, $(1, 4)$, $(1, 6)$, $(2, 3)$, $(2, 4)$, $(2, 6)$, $(5, 6)$ which total to 7 inversions which is the maximum possible.

F. Quests

3 seconds, 256 megabytes

There are n quests. If you complete the i-th quest, you will gain a_i coins. You can only complete at most one quest per day. However, once you complete a quest, you cannot do the same quest again for k days. (For example, if $k = 2$ and you do quest 1 on day 1, then you cannot do it on day 2 or 3, but you can do it again on day 4.)

You are given two integers c and d. Find the maximum value of k such that you can gain at least c coins over d days. If no such k exists, output `Impossible`. If k can be arbitrarily large, output `Infinity`.

Input

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

The first line of each test case contains three integers n, c, d ($2 \le n \le 2 \cdot 10^5$; $1 \le c \le 10^{16}$; $1 \le d \le 2 \cdot 10^5$) — the number of quests, the number of coins you need, and the number of days.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$) — the rewards for the quests.

The sum of n over all test cases does not exceed $2 \cdot 10^5$, and the sum of d over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output one of the following.

- If no such k exists, output `Impossible`.
- If k can be arbitrarily large, output `Infinity`.
- Otherwise, output a single integer — the maximum value of k such that you can gain at least c coins over d days.

Please note, the checker is **case-sensitive**, and you should output strings exactly as they are given.

input
6 2 5 4 1 2 2 20 10 100 10 3 100 3 7 2 6 4 20 3 4 5 6 7 4 100000000000 2022 8217734 927368 26389746 627896974 2 20 4 5 1

output
2 Infinity Impossible 1 12 0

In the first test case, one way to earn 5 coins over 4 days with $k = 2$ is as follows:

- Day 1: do quest 2, and earn 2 coins.
- Day 2: do quest 1, and earn 1 coin.
- Day 3: do nothing.
- Day 4: do quest 2, and earn 2 coins.

In total, we earned $2 + 1 + 2 = 5$ coins.
In the second test case, we can make over 20 coins on the first day itself by doing the first quest to earn 100 coins, so the value of k can be arbitrarily large, since we never need to do another quest.

In the third test case, no matter what we do, we can't earn 100 coins over 3 days.

G. SlavicG's Favorite Problem

2 seconds, 256 megabytes

You are given a weighted tree with n vertices. Recall that a tree is a connected graph without any cycles. A weighted tree is a tree in which each edge has a certain weight. The tree is undirected, it doesn't have a root.

Since trees bore you, you decided to challenge yourself and play a game on the given tree.

In a move, you can travel from a node to one of its neighbors (another node it has a direct edge with).

You start with a variable x which is initially equal to 0. When you pass through edge i, x changes its value to x XOR w_i (where w_i is the weight of the i-th edge).

Your task is to go from vertex a to vertex b, but you are allowed to enter node b if and only if after traveling to it, the value of x will become 0. In other words, you can travel to node b only by using an edge i such that x XOR $w_i = 0$. Once you enter node b the game ends and you win.

Additionally, you can teleport **at most once** at any point in time to any vertex except vertex b. You can teleport from any vertex, even from a.

Answer with "YES" if you can reach vertex b from a, and "NO" otherwise.

Note that XOR represents the bitwise XOR operation.

Input

The first line contains a single integer t ($1 \le t \le 1000$) — the number of test cases.

The first line of each test case contains three integers n, a, and b ($2 \le n \le 10^5$), ($1 \le a, b \le n; a \ne b$) — the number of vertices, and the starting and desired ending node respectively.

Each of the next $n - 1$ lines denotes an edge of the tree. Edge i is denoted by three integers u_i, v_i and w_i — the labels of vertices it connects ($1 \le u_i, v_i \le n; u_i \ne v_i; 1 \le w_i \le 10^9$) and the weight of the respective edge.

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

Output

For each test case output "YES" if you can reach vertex b, and "NO" otherwise.

input

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

output

```
YES
NO
YES
```

For the first test case, we can travel from node 1 to node 3, x changing from 0 to 1, then we travel from node 3 to node 2, x becoming equal to 3. Now, we can teleport to node 3 and travel from node 3 to node 4, reaching node b, since x became equal to 0 in the end, so we should answer "YES".

For the second test case, we have no moves, since we can't teleport to node b and the only move we have is to travel to node 2 which is impossible since x wouldn't be equal to 0 when reaching it, so we should answer "NO".