Educational Codeforces Round 118 (Rated for Div. 2)

A. Long Comparison

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

Monocarp wrote down two numbers on a whiteboard. Both numbers follow a specific format: a positive integer x with p zeros appended to its end.

Now Monocarp asks you to compare these two numbers. Can you help him?

Input

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

The first line of each testcase contains two integers x_1 and p_1 $(1 \le x_1 \le 10^6; 0 \le p_1 \le 10^6)$ — the description of the first number.

The second line of each testcase contains two integers x_2 and p_2 $(1 \le x_2 \le 10^6; 0 \le p_2 \le 10^6)$ — the description of the second number.

Output

For each testcase print the result of the comparison of the given two numbers. If the first number is smaller than the second one, print '<'. If the first number is greater than the second one, print '>'. If they are equal, print '='.

Example

input

```
5
2 1
19 0
10 2
100 1
1999 0
2 3
1 0
1 0
99 0
1 2
```

output

```
>
=
<
=
<
```

Note

The comparisons in the example are: $20 > 19$, $1000 = 1000$, $1999 < 2000$, $1 = 1$, $99 < 100$.

B. Absent Remainder

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

You are given a sequence $a_1, a_2, \dots, a_n$ consisting of n pairwise distinct positive integers.

Find $\left\lfloor \frac{n}{2} \right\rfloor$ different pairs of integers x and y such that:

- $x \ne y$;
- x and y appear in a;
- $x \ mod \ y$ doesn't appear in a.

Note that some x or y can belong to multiple pairs.

$\lfloor x \rfloor$ denotes the floor function — the largest integer less than or equal to x. $x \ mod \ y$ denotes the remainder from dividing x by y.

If there are multiple solutions, print any of them. It can be shown that at least one solution always exists.

Input

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

The first line of each testcase contains a single integer n ($2 \le n \le 2 \cdot 10^5$) — the length of the sequence.

The second line of each testcase contains n integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le 10^6$).

All numbers in the sequence are pairwise distinct. The sum of n over all testcases doesn't exceed $2 \cdot 10^5$.

Output

The answer for each testcase should contain $\left\lfloor \frac{n}{2} \right\rfloor$ different pairs of integers x and y such that $x \neq y$, x and y appear in a and $x \ mod \ y$ doesn't appear in a. Print the pairs one after another.

You can print the pairs in any order. However, the order of numbers in the pair should be exactly such that the first number is x and the second number is y. All pairs should be pairwise distinct.

If there are multiple solutions, print any of them.

Example

input

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

output

```
4 1
8 2
8 4
9 5
7 5
8 7
4 3
5 2
```

Note

In the first testcase there are only two pairs: $(1, 4)$ and $(4, 1)$. $\left\lfloor \frac{2}{2} \right\rfloor = 1$, so we have to find one pair. $1 \ mod \ 4 = 1$, and 1 appears in a, so that pair is invalid. Thus, the only possible answer is a pair $(4, 1)$.

In the second testcase, we chose pairs $8 \ mod \ 2 = 0$ and $8 \ mod \ 4 = 0$. 0 doesn't appear in a, so that answer is valid. There are multiple possible answers for that testcase.

In the third testcase, the chosen pairs are $9 \ mod \ 5 = 4$ and $7 \ mod \ 5 = 2$. Neither 4, nor 2, appears in a, so that answer is valid.

C. Poisoned Dagger

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

Monocarp is playing yet another computer game. In this game, his character has to kill a dragon. The battle with the dragon lasts 100^{500} seconds, during which Monocarp attacks the dragon with a poisoned dagger. The i-th attack is performed at the beginning of the a_i-th second from the battle start. The dagger itself does not deal damage, but it applies a poison effect on the dragon, which deals 1 damage during each of the next k seconds (starting with the same second when the dragon was stabbed by the dagger). However, if the dragon has already been poisoned, then the dagger updates the poison effect (i.e. cancels the current poison effect and applies a new one).

For example, suppose $k = 4$, and Monocarp stabs the dragon during the seconds 2, 4 and 10. Then the poison effect is applied at the start of the 2-nd second and deals 1 damage during the 2-nd and 3-rd seconds; then, at the beginning of the 4-th second, the poison effect is reapplied, so it deals exactly 1 damage during the seconds 4, 5, 6 and 7; then, during the 10-th second, the poison effect is applied again, and it deals 1 damage during the seconds 10, 11, 12 and 13. In total, the dragon receives 10 damage.

Monocarp knows that the dragon has h hit points, and if he deals at least h damage to the dragon during the battle — he slays the dragon. Monocarp has not decided on the strength of the poison he will use during the battle, so he wants to find the minimum possible value of k (the number of seconds the poison effect lasts) that is enough to deal at least h damage to the dragon.

Input

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

The first line of the test case contains two integers n and h ($1 \le n \le 100; 1 \le h \le 10^{18}$) — the number of Monocarp's attacks and the amount of damage that needs to be dealt.

The second line contains n integers a_1, a_2, ..., a_n ($1 \le a_i \le 10^9; a_i < a_{i+1}$), where a_i is the second when the i-th attack is performed.

Output

For each test case, print a single integer — the minimum value of the parameter k, such that Monocarp will cause at least h damage to the dragon.

Example

input

```
4
2 5
1 5
3 10
2 4 10
5 3
1 2 4 5 7
4 1000
3 25 64 1337
```

output

```
3
4
1
470
```

Note

In the first example, for $k = 3$, damage is dealt in seconds $[1, 2, 3, 5, 6, 7]$.

In the second example, for $k = 4$, damage is dealt in seconds $[2, 3, 4, 5, 6, 7, 10, 11, 12, 13]$.

In the third example, for $k = 1$, damage is dealt in seconds $[1, 2, 4, 5, 7]$.

D. MEX Sequences

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

Let's call a sequence of integers $x_1, x_2, \dots, x_k$ *MEX-correct* if for all i ($1 \le i \le k$) $|x_i - \operatorname{MEX}(x_1, x_2, \dots, x_i)| \le 1$ holds. Where $\operatorname{MEX}(x_1, \dots, x_k)$ is the minimum non-negative integer that doesn't belong to the set $x_1, \dots, x_k$. For example, $\operatorname{MEX}(1, 0, 1, 3) = 2$ and $\operatorname{MEX}(2, 1, 5) = 0$.

You are given an array a consisting of n non-negative integers. Calculate the number of non-empty *MEX-correct* subsequences of a given array. The number of subsequences can be very large, so print it modulo 998244353.

Note: a subsequence of an array a is a sequence $[a_{i_1}, a_{i_2}, \dots, a_{i_m}]$ meeting the constraints $1 \le i_1 < i_2 < \dots < i_m \le n$. If two different ways to choose the sequence of indices $[i_1, i_2, \dots, i_m]$ yield the same subsequence, the resulting subsequence should be counted twice (i. e. two subsequences are different if their sequences of indices $[i_1, i_2, \dots, i_m]$ are not the same).

Input

The first line contains a single integer t ($1 \le t \le 10^5$) — the number of test cases.

The first line of each test case contains a single integer n ($1 \le n \le 5 \cdot 10^5$).

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

The sum of n over all test cases doesn't exceed $5 \cdot 10^5$.

Output

For each test case, print a single integer — the number of non-empty *MEX-correct* subsequences of a given array, taken modulo 998244353.

Example

input

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

output

```
4
2
31
```

```
7
```

Note

In the first example, the valid subsequences are $[0]$, $[1]$, $[0,1]$ and $[0,2]$.

In the second example, the valid subsequences are $[0]$ and $[1]$.

In the third example, any non-empty subsequence is valid.

E. Crazy Robot

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

There is a grid, consisting of n rows and m columns. Each cell of the grid is either free or blocked. One of the free cells contains a lab. All the cells beyond the borders of the grid are also blocked.

A crazy robot has escaped from this lab. It is currently in some free cell of the grid. You can send one of the following commands to the robot: "move right", "move down", "move left" or "move up". Each command means moving to a neighbouring cell in the corresponding direction.

However, as the robot is crazy, it will do anything except following the command. Upon receiving a command, it will choose a direction such that it differs from the one in command and the cell in that direction is not blocked. If there is such a direction, then it will move to a neighbouring cell in that direction. Otherwise, it will do nothing.

We want to get the robot to the lab to get it fixed. For each free cell, determine if the robot can be forced to reach the lab starting in this cell. That is, after each step of the robot a command can be sent to a robot such that no matter what different directions the robot chooses, it will end up in a lab.

Input

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

The first line of each testcase contains two integers n and m $(1 \le n, m \le 10^6;\ n \cdot m \le 10^6)$ — the number of rows and the number of columns in the grid.

The i-th of the next n lines provides a description of the i-th row of the grid. It consists of m elements of one of three types:

- '.' — the cell is free;
- '#' — the cell is blocked;
- 'L' — the cell contains a lab.

The grid contains exactly one lab. The sum of $n \cdot m$ over all testcases doesn't exceed 10^6.

Output

For each testcase find the free cells that the robot can be forced to reach the lab from. Given the grid, replace the free cells (marked with a dot) with a plus sign ('+') for the cells that the robot can be forced to reach the lab from. Print the resulting grid.

Example

input

```
4
3 3
...
.L.
...
4 5
#....
..##L
...#.
.....
1 1
L
1 9
....L..#.
```

output

```
...
.L.
...
#++++
..##L
...#+
...++
L
++++L++#.
```

Note

In the first testcase there is no free cell that the robot can be forced to reach the lab from. Consider a corner cell. Given any direction, it will move to a neighbouring border grid that's not a corner. Now consider a non-corner free cell. No matter what direction you send to the robot, it can choose a different direction such that it ends up in a corner.

In the last testcase, you can keep sending the command that is opposite to the direction to the lab and the robot will have no choice other than move towards the lab.

F. Tree Coloring

time limit per test: 4.5 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

You are given a rooted tree consisting of n vertices numbered from 1 to n. The root of the tree is the vertex 1.

You have to color all vertices of the tree into n colors (also numbered from 1 to n) so that there is exactly one vertex for each color. Let c_i be the color of vertex i, and p_i be the parent of vertex i in the rooted tree. The coloring is considered beautiful if there is no vertex k ($k > 1$) such that $c_k = c_{p_k} - 1$, i. e. no vertex such that its color is less than the color of its parent by **exactly** 1.

Calculate the number of beautiful colorings, and print it modulo 998244353.

Input

The first line contains one integer n ($2 \le n \le 250000$) — the number of vertices in the tree.

Then $n - 1$ lines follow, the i-th line contains two integers x_i and y_i ($1 \le x_i, y_i \le n$; $x_i \ne y_i$) denoting an edge between the vertex x_i and the vertex y_i. These edges form a tree.

Output

Print one integer — the number of beautiful colorings, taken modulo 998244353.

Examples

input

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

output

```
42
```

input

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

output

```
53
```

input

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

output

```
955085064
```