Cool, It's Yesterday Four Times More

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 megabytes

After the great success in 2018, 2019, 2020, 2021 and 2022, Nanjing University of Aeronautics and Astronautics (NUAA) will host the *International Collegiate Programming Contest* (ICPC) Nanjing regional for the sixth time in a row.

Team ***Power of Two*** and team ***Three Hold Two*** won the champion title for Tsinghua University in 2018 and 2019. In 2020, 2021 and 2022, team ***Inverted Cross*** from Peking University won the three-peat champion titles. This year, there are around 330 teams participating in the contest. There are at most 33 gold medals, 66 silver medals and 99 bronze medals that will be awarded (note that these numbers are for reference only). We are looking forward to seeing participants' outstanding performance!

What's even better is that, as the pandemic has come to an end, we can finally gather in Nanjing to participate in this wonderful contest. We'd like to be grateful for the hard work done by all staff and volunteers for this contest. Thank you all for your great contribution to this contest!

The 2018 ICPC Asia Nanjing Regional Contest

In the 2018 contest, problem K, ***Kangaroo Puzzle***, requires the contestants to construct an operation sequence for the game:

> The puzzle is a grid with n rows and m columns ($1 \leq n, m \leq 20$) and there are some (at least 2) kangaroos standing in the puzzle. The player's goal is to control them to get together. There are some walls in some cells and the kangaroos cannot enter the cells with walls. The other cells are empty. The kangaroos can move from an empty cell to an adjacent empty cell in four directions: up, down, left, and right.
>
> There is exactly one kangaroo in every empty cell in the beginning and the player can control the kangaroos by pressing the button U, D, L, R on the keyboard. The kangaroos will move simultaneously according to the button you press.
>
> The contestant needs to construct an operating sequence of at most 5×10^4 steps consisting of U, D, L, R only to achieve the goal.

In the 2020 contest, problem A, ***Ah, It's Yesterday Once More***, requires the contestants to construct an input map to hack the following code of the problem described before:

```
#include <bits/stdc++.h>
using namespace std;
string s = "UDLR";
int main()
{
  srand(time(NULL));
  for (int i = 1; i <= 50000; i++) putchar(s[rand() % 4]);
  return 0;
}
```

In the 2021 contest, problem A, ***Oops, It's Yesterday Twice More***, also requires the contestants to construct an operation sequence for the game:

> This time, every cell in the grid stands exactly one kangaroo. You need to construct an operating sequence consisting only of characters 'U', 'D', 'L', and 'R'. After applying it, you must make sure every kangaroo will gather at the specific cell (a, b). The length of the operating sequence cannot exceed $3(n-1)$. As always, the kangaroos will move simultaneously according to the operation you command.

In the 2022 contest, problem A, ***Stop, Yesterday Please No More***, asks the contestants to solve the following counting problem:

> This time, every cell (except one which is a hole) in the grid stands exactly one kangaroo. The operating sequence is given and all kangaroos stepping out of the grid or onto the hole will be removed. Given the number of kangaroos remaining after all operations, count the number of positions which might be the hole.

Now, in the 2023 contest, the kangaroo problem is back again! We don't know why problem setters are so obsessed with kangaroos but the problem is as follows:

You are given a grid with n rows and m columns. Each cell is either a hole or empty. In each empty cell stands exactly one kangaroo.

Similarly, the kangaroos are controlled by pressing the button U, D, L, R on the keyboard. All kangaroos will move simultaneously according to the button pressed. Specifically, for any kangaroo located in the cell on the i-th row and the j-th column, indicated by (i, j):

1. Button U: it will move to $(i-1, j)$.
2. Button D: it will move to $(i+1, j)$.
3. Button L: it will move to $(i, j-1)$.
4. Button R: it will move to $(i, j+1)$.

If a kangaroo steps onto a hole or steps out of the grid, it will be removed from the grid. If after applying a sequence of operations (possibly an empty sequence) there is exactly one kangaroo remaining on the grid, that kangaroo becomes the winner.

The problem is: for each kangaroo, determine if there exists a sequence of operations to make it the winner. Output the total number of kangaroos which are possible to become the winner.

Input

There are multiple test cases. The first line of the input contains an integer T indicating the number of test cases. For each test case:

The first line contains two integers n and m ($1 \le n, m \le 10^3$, $1 \le n \times m \le 10^3$) indicating the number of rows and columns of the grid.

For the following n lines, the i-th line contains a string $s_{i,1}s_{i,2}\cdots s_{i,m}$ of length m where each character is either '.' (dot, ascii: 46) or 'O' (capitalized letter, ascii: 79). If $s_{i,j}$ is a '.' then grid (i, j) is empty; If $s_{i,j}$ is a 'O' then grid (i, j) is a hole.

It's guaranteed that the sum of $n \times m$ of all test cases will not exceed 5×10^3.

Output

For each test case output one line containing one integer indicating the number of kangaroos for which there exists a sequence of operations to make it the winner.

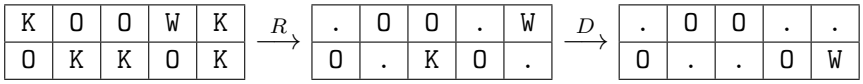
Example

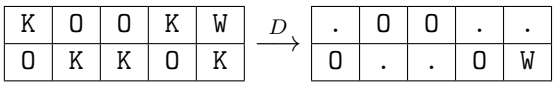
standard input	standard output
4 2 5 .OO.. O..O. 1 3 O.O 1 3 .O. 2 3 OOO OOO	3 1 0 0

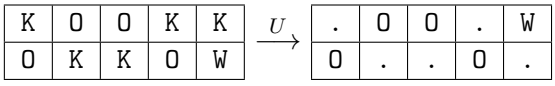
Note

The sample test cases are explained below. We use 'W' to indicate the kangaroo which later becomes winner and 'K' to indicate the other kangaroos.

For the first sample test case, kangaroos initially located at $(1, 4)$, $(1, 5)$ and $(2, 5)$ may become winners. Possible sequences of operations are shown as follows:







For the second sample test case, as there is only one kangaroo, no operation is needed for it to become the winner.

For the third sample test case, as any operation will remove the two kangaroos at the same time, there is no possible winner.

For the fourth sample test case, as there is no kangaroo, there is no possible winner.

[DivC] Contest #4

Statement is not available on English language

B. 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
```

C. Gift Carpet

1 second, 256 megabytes

Recently, Tema and Vika celebrated Family Day. Their friend Arina gave them a carpet, which can be represented as an $n \cdot m$ table of lowercase Latin letters.

Vika hasn't seen the gift yet, but Tema knows what kind of carpets she likes. Vika will like the carpet if she can read her name on. She reads column by column from left to right and chooses one or zero letters from current column.

Formally, the girl will like the carpet if it is possible to select four distinct columns in order from left to right such that the first column contains "v", the second one contains "i", the third one contains "k", and the fourth one contains "a".

Help Tema understand in advance whether Vika will like Arina's gift.

Input
Each test consists of multiple test cases. The first line of input contains a single integer t $(1 \le t \le 100)$ — the number of test cases. Then follows the description of the test cases.

The first line of each test case contains two integers n, m $(1 \le n, m \le 20)$ — the sizes of the carpet.

The next n lines contain m lowercase Latin letters each, describing the given carpet.

Output

Output
For each set of input data, output "YES" if Vika will like the carpet, 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.

input

```
5
1 4
vika
3 3
bad
car
pet
4 4
vvvv
iiii
kkkk
aaaa
4 4
vkak
iiai
avvk
viaa
4 7
vbickda
vbickda
vbickda
vbickda
```

output

```
YES
NO
YES
NO
YES
```

In the first sample, Vika can read her name from left to right.

In the second sample, Vika cannot read the character "v", so she will not like the carpet.

D. Teleportation

1 second, 1024 megabytes

Bobo recently visited a strange teleportation system. The system contains n rooms, numbered 0 through $n-1$. A teleporting device is installed in each room. Each teleporting device contains a dashboard that looks like a clock surface with a hand on it, showing numbers 0 through $n-1$ in clockwise order. Initially, the hand on the dashboard of the teleport device in the i-th $(0 \le i \le n-1)$ room points to the number a_i.

When Bobo is in room i $(0 \le i \le n-1)$, he may do the following operation any number of times:

- *Teleport.* Immediately teleport to the room $(i + a_i) \bmod n$.
- *Move the hand clockwise.* Set $a_i \leftarrow a_i + 1$.

Each operation takes one unit of time. Bobo starts at room 0, and he wants to reach some room x as quickly as possible. He wonders how long it is needed.

Input
The first line of input contains two integers n $(2 \le n \le 10^5)$ and x $(1 \le x \le n-1)$, denoting the number of rooms and Bobo's destination room, respectively.

The next line contains n integers $a_0, a_1, \ldots, a_{n-1}$ $(0 \le a_i \le n-1)$, where a_i $(0 \le i \le n-1)$ denotes the number the hand in the i-th room points to.

Output
Output an integer in a line, denoting the minimum time Bobo needs to reach room x from room 0.

input

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

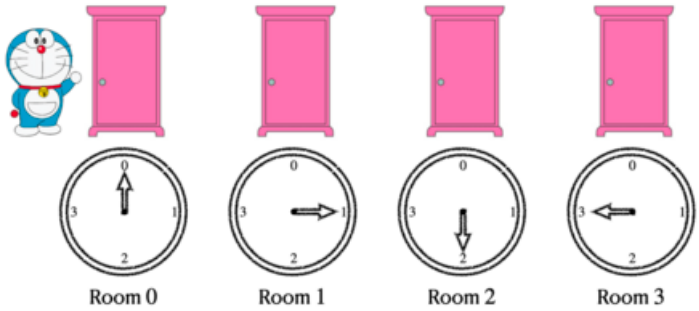
output

```
4
```

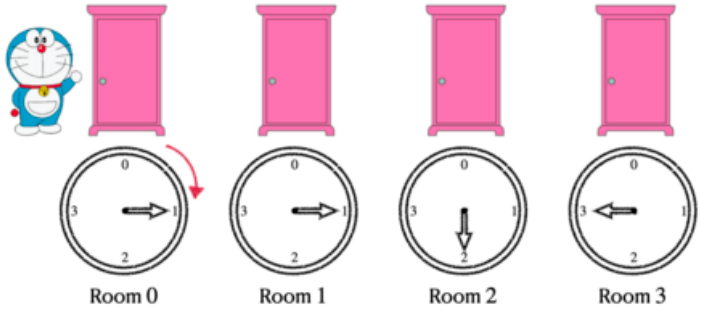
input
4 3 0 0 0 0
output
4

input
4 3 2 2 2 2
output
2

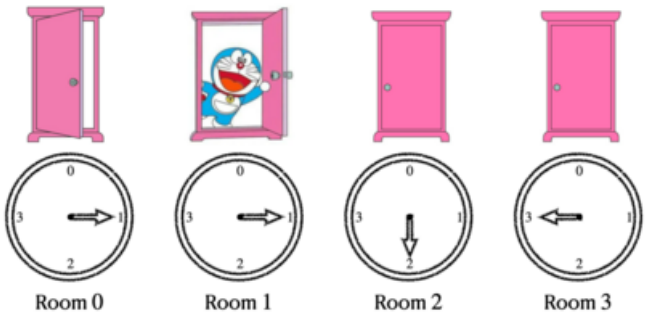
Here, we provide graphical illustrations of one possible optimal way in the first sample. Initially, Bobo is at room 0, and the hand on each dashboard is at $0, 1, 2, 3$, respectively.



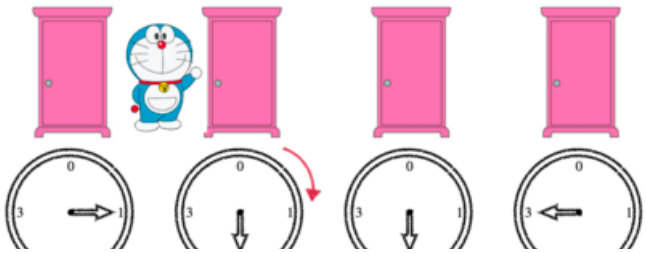
The first operation Bobo does is to move the hand clockwise so that the hand on the dashboard in room 0 points to 1. ($a_0 = 1$)



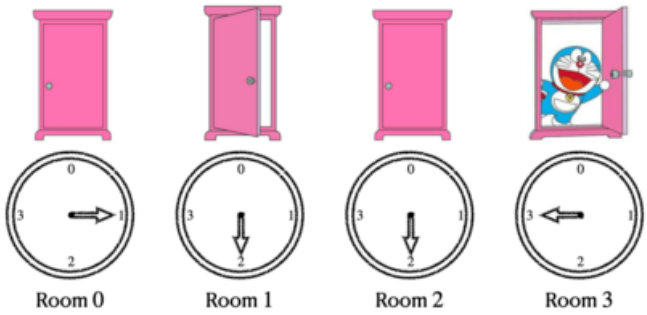
Then Bobo teleports to room $(0 + a_0) \bmod n = 1$.



After that, Bobo moves the hand clockwise so that the hand on the dashboard in room 1 points to 2. ($a_1 = 2$).



Then Bobo teleports to room $(1 + a_1) \bmod n = 3$, reaching his desired destination. It takes an overall of 4 operations.



E. A Hard Problem

1 second, 512 megabytes

Given a positive integer n, you need to find out the minimum integer k such that for any subset T of the set $\{1, 2, \cdots, n\}$ of size k, there exist two different integers $u, v \in T$ that u is a factor of v.

Input

The first line contains an integer T $(1 \le T \le 10^5)$ indicating the number of test cases.

Each of the following T lines contains an integer n $(2 \le n \le 10^9)$ describing a test case.

Output

For each test case, output a line containing an integer which indicates the answer.

input
4 2 3 4 5
output
2 3 3 4

F. Good Subarrays

2 seconds, 256 megabytes

You are given an array $a_1, a_2, \ldots, a_n$ consisting of integers from 0 to 9. A subarray $a_l, a_{l+1}, a_{l+2}, \ldots, a_{r-1}, a_r$ is good if the sum of elements of this subarray is equal to the length of this subarray ($\sum_{i=l}^{r} a_i = r - l + 1$).

For example, if $a = [1, 2, 0]$, then there are 3 good subarrays: $a_{1\ldots1} = [1]$, $a_{2\ldots3} = [2, 0]$ and $a_{1\ldots3} = [1, 2, 0]$.

Calculate the number of good subarrays of the array a.

Input

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

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

The second line of each test case contains a string consisting of n decimal digits, where the i-th digit is equal to the value of a_i.

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

Output

For each test case print one integer — the number of good subarrays of the array a.

input

3 3 120 5 11011 6 600005

output
3 6 1

The first test case is considered in the statement.

In the second test case, there are 6 good subarrays: $a_{1\ldots1}$, $a_{2\ldots2}$, $a_{1\ldots2}$, $a_{4\ldots4}$, $a_{5\ldots5}$ and $a_{4\ldots5}$.

In the third test case there is only one good subarray: $a_{2\ldots6}$.

G. The number on the board

2 seconds, 256 megabytes

Some natural number was written on the board. Its sum of digits was not less than k. But you were distracted a bit, and someone changed this number to n, replacing some digits with others. It's known that the length of the number didn't change.

You have to find the minimum number of digits in which these two numbers can differ.

Input

The first line contains integer k ($1 \le k \le 10^9$).

The second line contains integer n ($1 \le n < 10^{100000}$).

There are no leading zeros in n. It's guaranteed that this situation is possible.

Output

Print the minimum number of digits in which the initial number and n can differ.

input
3 11

output
1

input
3 99

output
0

In the first example, the initial number could be 12.

In the second example the sum of the digits of n is not less than k. The initial number could be equal to n.

H. Moamen and k-subarrays

2 seconds, 256 megabytes

Moamen has an array of n **distinct** integers. He wants to sort that array in non-decreasing order by doing the following operations in order **exactly once**:

1. Split the array into exactly k non-empty subarrays such that each element belongs to exactly one subarray.
2. Reorder these subarrays arbitrary.
3. Merge the subarrays in their new order.

A sequence a is a subarray of a sequence b if a can be obtained from b by deletion of several (possibly, zero or all) elements from the beginning and several (possibly, zero or all) elements from the end.

Can you tell Moamen if there is a way to sort the array in non-decreasing order using the operations written above?

Input
The first line contains a single integer t ($1 \le t \le 10^3$) — the number of test cases. The description of the test cases follows.

The first line of each test case contains two integers n and k ($1 \le k \le n \le 10^5$).

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le |a_i| \le 10^9$). It is guaranteed that all numbers are **distinct**.

It is guaranteed that the sum of n over all test cases does not exceed $3 \cdot 10^5$.

Output
For each test case, you should output a single string.

If Moamen can sort the array in non-decreasing order, output "YES" (without quotes). Otherwise, output "NO" (without quotes).

You can print each letter of "YES" and "NO" in any case (upper or lower).

input
3 5 4 6 3 4 2 1 4 2 1 -4 0 -2 5 1 1 2 3 4 5
output
Yes No Yes

In the first test case, $a = [6, 3, 4, 2, 1]$, and $k = 4$, so we can do the operations as follows:

1. Split a into $\{[6], [3, 4], [2], [1]\}$.
2. Reorder them: $\{[1], [2], [3, 4], [6]\}$.
3. Merge them: $[1, 2, 3, 4, 6]$, so now the array is sorted.

In the second test case, there is no way to sort the array by splitting it into only 2 subarrays.

As an example, if we split it into $\{[1, -4], [0, -2]\}$, we can reorder them into $\{[1, -4], [0, -2]\}$ or $\{[0, -2], [1, -4]\}$. However, after merging the subarrays, it is impossible to get a sorted array.

I. String Reconstruction

2 seconds, 256 megabytes

Ivan had string s consisting of small English letters. However, his friend Julia decided to make fun of him and hid the string s. Ivan preferred making a new string to finding the old one.

Ivan knows some information about the string s. Namely, he remembers, that string t_i occurs in string s at least k_i times or more, he also remembers exactly k_i positions where the string t_i occurs in string s: these positions are $x_{i,1}, x_{i,2}, \ldots, x_{i,k_i}$. He remembers n such strings t_i.

You are to reconstruct **lexicographically minimal** string s such that it fits all the information Ivan remembers. Strings t_i and string s consist of small English letters only.

Input
The first line contains single integer n ($1 \le n \le 10^5$) — the number of strings Ivan remembers.

The next n lines contain information about the strings. The i-th of these lines contains non-empty string t_i, then positive integer k_i, which equal to the number of times the string t_i occurs in string s, and then k_i distinct positive integers $x_{i,1}, x_{i,2}, \ldots, x_{i,k_i}$ in increasing order — positions, in which occurrences of the string t_i in the string s start. It is guaranteed that the sum of lengths of strings t_i doesn't exceed 10^6, $1 \le x_{i,j} \le 10^6$, $1 \le k_i \le 10^6$, and the sum of all k_i doesn't exceed 10^6. The strings t_i can coincide.

It is guaranteed that the input data is not self-contradictory, and thus at least one answer **always** exists.

Output
Print lexicographically minimal string that fits all the information Ivan remembers.

input
3

```
a 4 1 3 5 7
ab 2 1 5
ca 1 4
```

output

```
abacaba
```

input

```
1
a 1 3
```

output

```
aaa
```

input

```
3
ab 1 1
aba 1 3
ab 2 3 5
```

output

```
ababab
```

J. Fake NP

1 second, 256 megabytes

Tavak and Seyyed are good friends. Seyyed is very funny and he told Tavak to solve the following problem instead of *longest-path*.

You are given l and r. For all integers from l to r, inclusive, we wrote down all of their integer divisors except 1. Find the integer that we wrote down the maximum number of times.

Solve the problem to show that it's not a *NP* problem.

Input

The first line contains two integers l and r ($2 \le l \le r \le 10^9$).

Output

Print single integer, the integer that appears maximum number of times in the divisors.

If there are multiple answers, print any of them.

input

```
19 29
```

output

```
2
```

input

```
3 6
```

output

```
3
```

Definition of a divisor: https://www.mathsisfun.com/definitions/divisor-of-an-integer-.html

The first example: from 19 to 29 these numbers are divisible by 2: $\{20, 22, 24, 26, 28\}$.

The second example: from 3 to 6 these numbers are divisible by 3: $\{3, 6\}$.

K. Game on Permutation

2 seconds, 256 megabytes

Alice and Bob are playing a game. They have a permutation p of size n (a permutation of size n is an array of size n where each element from 1 to n occurs exactly once). They also have a chip, which can be placed on any element of the permutation.

Alice and Bob make alternating moves: Alice makes the first move, then Bob makes the second move, then Alice makes the third move, and so on. During the first move, Alice chooses any element of the permutation and places the chip on that element. During each of the next moves, the current

player **has to** move the chip to any element that is simultaneously to the left and strictly less than the current element (i.e. if the chip is on the i-th element, it can be moved to the j-th element if $j < i$ and $p_j < p_i$). If a player cannot make a move (it is impossible to move the chip according to the rules of the game), that player **wins** the game.

Let's say that the i-th element of the permutation is **lucky** if the following condition holds:

- if Alice places the chip on the i-th element during her first move, she can win the game no matter how Bob plays (i.e. she has a winning strategy).

You have to calculate the number of lucky elements in the permutation.

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 ($1 \le n \le 3 \cdot 10^5$) – the number of elements in the permutation.

The second line contains n integers $p_1, p_2, \ldots, p_n$ ($1 \le p_i \le n$). All p_i are distinct.

The sum of n over all test cases doesn't exceed $3 \cdot 10^5$.

Output

For each test case, print a single integer — the number of lucky elements in the permutation.

input
4 3 2 1 3 2 2 1 3 1 2 3 4 2 1 4 3

output
1 0 1 2

In the first test case of the example, the 3-rd element of the permutation is lucky.

In the second test case of the example, there are no lucky elements.

In the third test case of the example, the 2-nd element of the permutation is lucky.

In the fourth test case of the example, the 3-rd and the 4-th element of the permutation are lucky.

L. Karen and Coffee

2.5 seconds, 512 megabytes

To stay woke and attentive during classes, Karen needs some coffee!

Karen, a coffee aficionado, wants to know the optimal temperature for brewing the perfect cup of coffee. Indeed, she has spent some time reading several recipe books, including the universally acclaimed "The Art of the Covfefe".

She knows n coffee recipes. The i-th recipe suggests that coffee should be brewed between l_i and r_i degrees, inclusive, to achieve the optimal taste.

Karen thinks that a temperature is *admissible* if at least k recipes recommend it.

Karen has a rather fickle mind, and so she asks q questions. In each question, given that she only wants to prepare coffee with a temperature between a and b, inclusive, can you tell her how many admissible integer temperatures fall within the range?

Input

The first line of input contains three integers, n, k ($1 \le k \le n \le 200000$), and q ($1 \le q \le 200000$), the number of recipes, the minimum number of recipes a certain temperature must be recommended by to be admissible, and the number of questions Karen has, respectively.

The next n lines describe the recipes. Specifically, the i-th line among these contains two integers l_i and r_i ($1 \le l_i \le r_i \le 200000$), describing that the i-th recipe suggests that the coffee be brewed between l_i and r_i degrees, inclusive.

The next q lines describe the questions. Each of these lines contains a and b, ($1 \le a \le b \le 200000$), describing that she wants to know the number of admissible integer temperatures between a and b degrees, inclusive.

Output

For each question, output a single integer on a line by itself, the number of admissible integer temperatures between a and b degrees, inclusive.

input

```
3 2 4
91 94
92 97
97 99
92 94
93 97
95 96
90 100
```

output

```
3
3
0
4
```

input

```
2 1 1
1 1
200000 200000
90 100
```

output

```
0
```

In the first test case, Karen knows 3 recipes.

1. The first one recommends brewing the coffee between 91 and 94 degrees, inclusive.
2. The second one recommends brewing the coffee between 92 and 97 degrees, inclusive.
3. The third one recommends brewing the coffee between 97 and 99 degrees, inclusive.

A temperature is *admissible* if at least 2 recipes recommend it.

She asks 4 questions.

In her first question, she wants to know the number of admissible integer temperatures between 92 and 94 degrees, inclusive. There are 3: 92, 93 and 94 degrees are all admissible.

In her second question, she wants to know the number of admissible integer temperatures between 93 and 97 degrees, inclusive. There are 3: 93, 94 and 97 degrees are all admissible.

In her third question, she wants to know the number of admissible integer temperatures between 95 and 96 degrees, inclusive. There are none.

In her final question, she wants to know the number of admissible integer temperatures between 90 and 100 degrees, inclusive. There are 4: 92, 93, 94 and 97 degrees are all admissible.

In the second test case, Karen knows 2 recipes.

1. The first one, "wikiHow to make Cold Brew Coffee", recommends brewing the coffee at exactly 1 degree.

2. The second one, "What good is coffee that isn't brewed at at least 36.3306 times the temperature of the surface of the sun?", recommends brewing the coffee at exactly 200000 degrees.

A temperature is *admissible* if at least 1 recipe recommends it.

In her first and only question, she wants to know the number of admissible integer temperatures that are actually reasonable. There are none.
