Codeforces Round #725 (Div. 3)

A. Stone Game

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

Polycarp is playing a new computer game. This game has n stones in a row. The stone on the position i has integer power a_i. **The powers of all stones are distinct**.

Each turn Polycarp can destroy either stone on the first position or stone on the last position (in other words, either the leftmost or the rightmost stone). When Polycarp destroys the stone it does not exist any more.

Now, Polycarp wants two achievements. He gets them if he destroys the stone with the **least** power and the stone with the **greatest** power. Help Polycarp find out what is the minimum number of moves he should make in order to achieve his goal.

For example, if $n = 5$ and $a = [1, 5, 4, 3, 2]$, then Polycarp could make the following moves:

- Destroy the leftmost stone. After this move $a = [5, 4, 3, 2]$;
- Destroy the rightmost stone. After this move $a = [5, 4, 3]$;
- Destroy the leftmost stone. After this move $a = [4, 3]$. Polycarp destroyed the stones with the greatest and least power, so he can end the game.

Please note that in the example above, you can complete the game in two steps. For example:

- Destroy the leftmost stone. After this move $a = [5, 4, 3, 2]$;
- Destroy the leftmost stone. After this move $a = [4, 3, 2]$. Polycarp destroyed the stones with the greatest and least power, so he can end the game.

Input

The first line contains an integer t ($1 \le t \le 100$). Then t test cases follow.

The first line of each test case contains one integer n ($2 \le n \le 100$) — the number of stones.

The second line contains n distinct integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$) — the power of the stones.

Output

For each test case, output the minimum number of moves required to destroy the stones with the greatest and the lowest power.

Example

input
5 5 1 5 4 3 2 8 2 1 3 4 5 6 8 7 8 4 2 3 1 8 6 7 5 4 3 4 2 1 4 2 3 1 4
output
2 4 5 3 2

B. Friends and Candies

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

Polycarp has n friends, the i-th of his friends has a_i candies. Polycarp's friends do not like when they have different numbers of candies. In other words they want all a_i to be the same. To solve this, Polycarp performs the following set of actions exactly **once**:

- Polycarp chooses k ($0 \le k \le n$) arbitrary friends (let's say he chooses friends with indices $i_1, i_2, \ldots, i_k$);

- Polycarp distributes their $a_{i_1} + a_{i_2} + \ldots + a_{i_k}$ candies among all n friends. During distribution for each of $a_{i_1} + a_{i_2} + \ldots + a_{i_k}$ candies he chooses new owner. That can be any of n friends. Note, that any candy can be given to the person, who has owned that candy before the distribution process.

Note that the number k is not fixed in advance and can be arbitrary. Your task is to find the minimum value of k.

For example, if $n = 4$ and $a = [4, 5, 2, 5]$, then Polycarp could make the following distribution of the candies:

- Polycarp chooses $k = 2$ friends with indices $i = [2, 4]$ and distributes $a_2 + a_4 = 10$ candies to make $a = [4, 4, 4, 4]$ (two candies go to person 3).

Note that in this example Polycarp cannot choose $k = 1$ friend so that he can redistribute candies so that in the end all a_i are equal.

For the data n and a, determine the **minimum** value k. With this value k, Polycarp should be able to select k friends and redistribute their candies so that everyone will end up with the same number of candies.

Input
The first line contains one integer t ($1 \le t \le 10^4$). Then t test cases follow.

The first line of each test case contains one integer n ($1 \le n \le 2 \cdot 10^5$).

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 10^4$).

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 minimum value of k, such that Polycarp can choose exactly k friends so that he can redistribute the candies in the desired way;
- "-1" if no such value k exists.

Example

input
5 4 4 5 2 5 2 0 4 5 10 8 5 1 4 1 10000 7 1 1 1 1 1 1 1
output
2 1 -1 0 0

C. Number of Pairs

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

You are given an array a of n integers. Find the number of pairs (i, j) ($1 \le i < j \le n$) where the sum of $a_i + a_j$ is greater than or equal to l and less than or equal to r (that is, $l \le a_i + a_j \le r$).

For example, if $n = 3$, $a = [5, 1, 2]$, $l = 4$ and $r = 7$, then two pairs are suitable:

- $i = 1$ and $j = 2$ ($4 \le 5 + 1 \le 7$);
- $i = 1$ and $j = 3$ ($4 \le 5 + 2 \le 7$).

Input
The first line contains an integer t ($1 \le t \le 10^4$). Then t test cases follow.

The first line of each test case contains three integers n, l, r ($1 \le n \le 2 \cdot 10^5$, $1 \le l \le r \le 10^9$) — the length of the array and the limits on the sum in the pair.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$).

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

Output
For each test case, output a single integer — the number of index pairs (i, j) ($i < j$), such that $l \le a_i + a_j \le r$.

Example

input
4 3 4 7 5 1 2 5 5 8 5 1 2 4 3 4 100 1000 1 1 1 1 5 9 13 2 5 5 1 1

output
2 7 0 1

D. Another Problem About Dividing Numbers

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

You are given two integers a and b. In one turn, you can do one of the following operations:

- Take an integer c ($c > 1$ and a **should be divisible by** c) and replace a with $\frac{a}{c}$;
- Take an integer c ($c > 1$ and b **should be divisible by** c) and replace b with $\frac{b}{c}$.

Your goal is to make a equal to b using exactly k turns.

For example, the numbers $a = 36$ and $b = 48$ can be made equal in 4 moves:

- $c = 6$, divide b by $c \Rightarrow a = 36, b = 8$;
- $c = 2$, divide a by $c \Rightarrow a = 18, b = 8$;
- $c = 9$, divide a by $c \Rightarrow a = 2, b = 8$;
- $c = 4$, divide b by $c \Rightarrow a = 2, b = 2$.

For the given numbers a and b, determine whether it is possible to make them equal using exactly k turns.

Input
The first line contains one integer t ($1 \le t \le 10^4$). Then t test cases follow.

Each test case is contains three integers a, b and k ($1 \le a, b, k \le 10^9$).

Output
For each test case output:

- "Yes", if it is possible to make the numbers a and b equal in **exactly** k turns;
- "No" otherwise.

The strings "Yes" and "No" can be output in any case.

Example

input
8 36 48 2 36 48 3 36 48 4 2 8 1 2 8 2 1000000000 1000000000 1000000000 1 2 1 2 2 1

output
YES YES YES YES YES NO YES NO

E. Funny Substrings

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

Polycarp came up with a new programming language. There are only two types of statements in it:

- "`x := s`": assign the variable named `x` the value `s` (where `s` is a string). For example, the statement `var := hello` assigns the variable named `var` the value `hello`. Note that `s` is the value of a string, not the name of a variable. Between the variable name, the `:=` operator and the string contains exactly one space each.
- "`x = a + b`": assign the variable named `x` the concatenation of values of two variables `a` and `b`. For example, if the program consists of three statements `a := hello`, `b := world`, `c = a + b`, then the variable `c` will contain the string `helloworld`. It is guaranteed that the program is correct and the variables `a` and `b` were previously defined. There is exactly one space between the variable names and the = and + operators.

All variable names and strings only consist of lowercase letters of the English alphabet and do not exceed 5 characters.

The result of the program is the number of occurrences of string `haha` in the string that was written to the variable in the last statement.

Polycarp was very tired while inventing that language. He asks you to implement it. Your task is — for given program statements calculate the number of occurrences of string `haha` in the last assigned variable.

Input

The first line contains an integer t ($1 \le t \le 10^3$). Then t test cases follow.

The first line of each test case contains a single integer n ($1 \le n \le 50$) — the number of statements in the program. All variable names and strings are guaranteed to consist only of lowercase letters of the English alphabet and do not exceed 5 characters.

This is followed by n lines describing the statements in the format described above. It is guaranteed that the program is correct.

Output

For each set of input data, output the number of occurrences of the `haha` substring in the string that was written to the variable in the last statement.

Example

input
4 6 a := h b := aha c = a + b c = c + c e = c + c d = a + c 15 x := haha x = x + x x = x + x x = x + x x = x + x x = x + x x = x + x x = x + x x = x + x x = x + x x = x + x x = x + x x = x + x x = x + x x = x + x 1 haha := hah 5 haahh := aaaha ahhhh = haahh + haahh haahh = haahh + haahh ahhhh = ahhhh + haahh ahhaa = haahh + ahhhh
output
3 32767 0 0

Note

In the first test case the resulting value of `d` is `hhahahaha`.

F. Interesting Function

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

You are given two integers l and r, where $l < r$. We will add 1 to l until the result is equal to r. Thus, there will be exactly $r - l$ additions performed. For each such addition, let's look at the number of digits that will be changed after it.

For example:

- if $l = 909$, then adding one will result in 910 and 2 digits will be changed;
- if you add one to $l = 9$, the result will be 10 and 2 digits will also be changed;
- if you add one to $l = 489999$, the result will be 490000 and 5 digits will be changed.

Changed digits always form a suffix of the result written in the decimal system.

Output the total number of changed digits, if you want to get r from l, adding 1 each time.

Input
The first line contains an integer t ($1 \le t \le 10^4$). Then t test cases follow.

Each test case is characterized by two integers l and r ($1 \le l < r \le 10^9$).

Output
For each test case, calculate the total number of changed digits if you want to get r from l, adding one each time.

Example

input
4 1 9 9 10 10 20 1 1000000000
output
8 2 11 1111111110

G. Gift Set

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

Polycarp has x of red and y of blue candies. Using them, he wants to make gift sets. Each gift set contains either a red candies and b blue candies, or a blue candies and b red candies. Any candy can belong to at most one gift set.

Help Polycarp to find the largest number of gift sets he can create.

For example, if $x = 10$, $y = 12$, $a = 5$, and $b = 2$, then Polycarp can make three gift sets:

- In the first set there will be 5 red candies and 2 blue candies;
- In the second set there will be 5 blue candies and 2 red candies;
- In the third set will be 5 blue candies and 2 red candies.

Note that in this example there is one red candy that Polycarp does not use in any gift set.

Input
The first line contains an integer t ($1 \le t \le 10^4$). Then t test cases follow.

Each test case consists of a single string containing four integers x, y, a, and b ($1 \le x, y, a, b \le 10^9$).

Output
For each test case, output one number — the maximum number of gift sets that Polycarp can make.

Example

input
9 10 12 2 5 1 1 2 2 52 311 13 27 1000000000 1000000000 1 1 1000000000 1 1 1000000000 1 1000000000 1000000000 1 1 2 1 1

```
7 8 1 2
4 1 2 3
```

output

```
3
0
4
1000000000
1
1
1
5
0
```