IECSE Coding Bootcamp Week 1

A. Ankit's String

1 s., 256 MB

Ankit is given a string s of length n.

It consists of lowercase letters of the alphabet.

He is given a task to chose some number k between 0 and n.

After that, he needs to select k characters of s and permute the characters the way he wants to, keeping the positions of the other $n-k$ characters remain unchanged.

He can perform the above-mentioned operation only once.

You need to determine the minimum value of k that Ankit requires to sort s alphabetically.

Input

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

The first line of each test case contains one integer n ($1 \leq n \leq 40$) — the length of the string.

The second line of each test case contains the string s. It is guaranteed that s contains only lowercase letters of the English alphabet.

Output

For each test case, output the minimum k that Ankit requires to obtain an alphabetically sorted string performing the above-mentioned operation.

input

```
4
3
lol
10
codeforces
5
aaaaa
4
dcba
```

output

```
2
6
0
4
```

In the **first test case**, we can choose the $k=2$ characters `"_ol"` and rearrange them as `"_lo"` (so the resulting string is `"llo"`). It is not possible to sort the string choosing strictly less than 2 characters.

In the **second test case**, one possible way to sort s is to consider the $k=6$ characters `"_o__force_"` and rearrange them as `"_c__efoor_"` (so the resulting string is `"ccdeefoors"`). One can show that it is not possible to sort the string choosing strictly less than 6 characters.

In the **third test case**, string s is already sorted (so we can choose $k=0$ characters).

In the **fourth test case**, we can choose all $k=4$ characters `"dcba"` and reverse the whole string (so the resulting string is `"abcd"`).

B. Find Same Differences

2 s., 256 MB

You are given an array a of n integers. Find out the number of indices (i, j) such that $i < j$ and $a_j - a_i = j - i$.

Input

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

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

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

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 number of pairs of indices (i, j) such that $i < j$ and $a_j - a_i = j - i$.

input

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

output

```
1
3
3
10
```

C. Shuffling

2 s., 256 MB

You are given an array a of n integers. Array is nice if for each pair of indexes $i < j$ the condition $j - a_j \neq i - a_i$ holds. Can you shuffle this array so that it becomes good? To shuffle an array means to reorder its elements arbitrarily (leaving the initial order is also an option).

For example, if $a = [1, 1, 3, 5]$, then shuffled arrays $[1, 3, 5, 1]$, $[3, 5, 1, 1]$ and $[5, 3, 1, 1]$ are good, but shuffled arrays $[3, 1, 5, 1]$, $[1, 1, 3, 5]$ and $[1, 1, 5, 3]$ aren't.

It's guaranteed that it's always possible to shuffle an array to meet this condition.

Input

The first line contains one integer t ($1 \leq t \leq 100$) — the number of test cases.

The first line of each test case contains one integer n ($1 \leq n \leq 100$) — the length of array a.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 100$).

Output

For each test case, print the shuffled version of the array a which is good.

input

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

output

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

D. Substring Balancing

2 s., 256 MB

Harry is given a string s of length n consisting of characters a and/or b.

Given $\mathrm{AB}(s)$ be the number of times the string ab in s as a **substring**. Similarly, $\mathrm{BA}(s)$ is the number of occurrences of ba in s as a **substring**.

Harry is given the freedom to choose any index i and replace s_i with character a or b.

What is the minimum number of steps that Harry would need to make to achieve $\mathrm{AB}(s) = \mathrm{BA}(s)$?

Reminder:

The number of occurrences of string d in s as substring is the number of indices i ($1 \le i \le |s| - |d| + 1$) such that substring $s_i s_{i+1} \dots s_{i+|d|-1}$ is equal to d. For example, $\mathrm{AB}(\mathtt{aabbbabaa}) = 2$ since there are two indices i: $i = 2$ where a<u>ab</u>bbabaa and $i = 6$ where aabbb<u>ab</u>aa.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 1000$). Description of the test cases follows.

The first and only line of each test case contains a single string s ($1 \le |s| \le 100$, where $|s|$ is the length of the string s), consisting only of characters a and/or b.

Output

For each test case, print the resulting string s with $\mathrm{AB}(s) = \mathrm{BA}(s)$ you'll get making the minimum number of steps.

If there are multiple answers, print any of them.

input

```
4
b
aabbbabaa
abbb
abbaab
```

output

```
b
aabbbabaa
bbbb
abbaaa
```

In the first test case, both $\mathrm{AB}(s) = 0$ and $\mathrm{BA}(s) = 0$ (there are no occurrences of ab (ba) in b), so can leave s untouched.

In the second test case, $\mathrm{AB}(s) = 2$ and $\mathrm{BA}(s) = 2$, so you can leave s untouched.

In the third test case, $\mathrm{AB}(s) = 1$ and $\mathrm{BA}(s) = 0$. For example, we can change s_1 to b and make both values zero.

In the fourth test case, $\mathrm{AB}(s) = 2$ and $\mathrm{BA}(s) = 1$. For example, we can change s_6 to a and make both values equal to 1.

E. We Love Odd

3 s., 256 MB

There are n positive integers $a_1, a_2, \dots, a_n$. For one move you can choose any even value c and divide by two **all** elements that equal c.

For example, if $a = [6, 8, 12, 6, 3, 12]$ and you choose $c = 6$, and a is transformed into $a = [3, 8, 12, 3, 3, 12]$ after the move.

You need to find the minimal number of moves for transforming a to an array of only odd integers (each element shouldn't be divisible by 2).

Input

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

The first line of a test case contains n ($1 \le n \le 2 \cdot 10^5$) — the number of integers in the sequence a. The second line contains positive integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le 10^9$).

The sum of n for all test cases in the input doesn't exceed $2 \cdot 10^5$.

Output

For t test cases print the answers in the order of test cases in the input. The answer for the test case is the minimal number of moves needed to make **all** numbers in the test case odd (i.e. not divisible by 2).

input

```
4
6
40 6 40 3 20 1
1
1024
4
2 4 8 16
3
3 1 7
```

output

```
4
10
4
0
```

In the first test case of the example, the optimal sequence of moves can be as follows:

- before making moves $a = [40, 6, 40, 3, 20, 1]$;
- choose $c = 6$;
- now $a = [40, 3, 40, 3, 20, 1]$;
- choose $c = 40$;
- now $a = [20, 3, 20, 3, 20, 1]$;
- choose $c = 20$;
- now $a = [10, 3, 10, 3, 10, 1]$;
- choose $c = 10$;
- now $a = [5, 3, 5, 3, 5, 1]$ — all numbers are odd.

Thus, all numbers became odd after 4 moves. In 3 or fewer moves, you cannot make them all odd.