Sherlock and Anagrams

Two strings are *anagrams* of each other if the letters of one string can be rearranged to form the other string. Given a string, find the number of pairs of substrings of the string which are anagrams of each other.

For example $s = abba$, the list of all anagrammatic pairs are $[a, a]$, $[ab, ba]$, $[b, b]$ and $[abb, bba]$ at positions $[[0], [3]]$, $[[0, 1], [2, 3]]$, $[[1], [2]]$ and $[[0, 1, 2], [1, 2, 3]]$ respectively.

Function Description

Complete the function *sherlockAndAnagrams* in the editor below. It must return an integer representing the number of anagrammatic substrings in s.

sherlockAndAnagrams has the following parameter(s):

- *s*: a string .

Input Format

The first line contains an integer q, the number of queries.
Each of the next q lines contains a string s to analyze.

Constraints

$1 \leq q \leq 10$
$2 \leq |s| \leq 100$
String s contains only the lowercase letters, ascii[a-z].

Output Format

For each query, print the number of unordered anagrammatic pairs on one line.

Sample Input 0

```
2
abba
abcd
```

Sample Output 0

```
4
0
```

Explanation 0

The list of all anagrammatic pairs is $[a, a]$, $[ab, ba]$, $[b, b]$ and $[abb, bba]$ at positions $[[0], [3]]$, $[[0, 1], [2, 3]]$, $[[1], [2]]$ and $[[0, 1, 2], [1, 2, 3]]$ respectively.

No anagrammatic pairs exist in the second query as no character repeats.

Sample Input 1

```
2
ifailuhkqq
kkkk
```

Sample Output 1

```
3
10
```

Explanation 1

For the first query, we have anagram pairs $[i, i]$, $[q, q]$ and $[ifa, fai]$ at positions $[[0], [3]]$, $[[8], [9]]$ and $[[0, 1, 2], [1, 2, 3]]$ respectively.

For the second query:
There are 6 anagrams of the form $[k, k]$ at positions $[[0], [1]]$, $[[0], [2]]$, $[[0], [3]]$, $[[1], [2]]$, $[[1], [3]]$ and $[[2], [3]]$.
There are 3 anagrams of the form $[kk, kk]$ at positions $[[0, 1], [1, 2]]$, $[[0, 1], [2, 3]]$ and $[[1, 2], [2, 3]]$.
There is 1 anagram of the form $[kkk, kkk]$ at position $[[0, 1, 2], [1, 2, 3]]$.