I split U in 3 (abc)

Valerio just found T strings $U_0, ..., U_{T-1}$, the i-th of which consists of N_i latin lowercase characters.

Since Valerio is very curious he asks you, for each $0 \leq i < T$, in how many ways one can split U_i into 3 possibly empty strings A, B, C such that $U_i = A + B + C$ under the constraints of each of the following scenarios:

- Scenario `abc`: the splits must satisfy $A \preceq B \preceq C$;
- Scenario `acb`: the splits must satisfy $A \preceq C \preceq B$;
- Scenario `bac`: the splits must satisfy $B \preceq A \preceq C$;
- Scenario `bca`: the splits must satisfy $B \preceq C \preceq A$;
- Scenario `cab`: the splits must satisfy $C \preceq A \preceq B$;
- Scenario `cba`: the splits must satisfy $C \preceq B \preceq A$.

where $+$ denotes string concatenation and $\preceq$ is the lexicographical less-than-or-equal-to.[1]

Implementation

You must submit a single file with the extension `.cpp`.

⇦ Among the attachments for this task, you will find a template `abc.cpp` with an example implementation.

⚠ A single input file may contain multiple testcases! Make sure to reset global variable between different runs.

You must implement the following function:

C++

```
void split(int N, string U,
           long long &abc, long long &acb, long long &bac,
           long long &bca, long long &cab, long long &cba);
```

- The integer N represents the length of the string U.
- The string U is one of the strings Valerio has found.
- The function should answer each scenario by assigning values to the corresponding parameter.
- This function is called T times during the execution of your program.

The grader will call the functions and print the returned values to the output file.

Sample Grader

A simplified version of the grader used during the correction is available in the directory related to this problem. You can use it to test your solutions locally. The sample grader reads input data from `stdin`, calls the function you need to implement, and writes to `stdout` in the following format.

[1] Formally, given two strings S and T, we have $S \preceq T$ if and only if one of the following is true:
- S is the empty string;
- Neither string is empty, and the first character of S comes before the first character of T in the latin alphabet.
- Neither string is empty, the first characters of the two strings are the same and $S' \preceq T'$ where S' and T' are the strings obtained by removing the first character from S and T respectively.

The input file is made up of $T + 1$ lines, where T is the number of test cases, containing:

- Line 1: an integer T.
- Line $2 + i$ $(0 \leq i < T)$: a string U_i.

The output file is made up of T lines, containing:

- Line $1 + i$ $(0 \leq i < T)$: the 6 answers your program gave of the i-th testcase, in the same order as they are presented in the statement.

Constraints

- The total length of the strings in an input case is at most 400 000.
- Each of the strings is nonempty and made up of lowercase latin characters.

Scoring

Your program will be tested on several test cases grouped into subtasks. The score for a subtask is equal to the worst score obtained on one of its test cases, multiplied by the value of the subtask.

The score for a test case depends on how many of the six scenarios you solve correctly, according to the following table:

Solved scenarios	0	1	2	3	4	5	6
Points	0	0.3	0.5	0.7	0.8	0.9	1

- **Subtask 0** [**0 points**]: Sample cases.
- **Subtask 1** [**10 points**]: The only character in the string is `a`.
- **Subtask 2** [**10 points**]: The total length of the strings in an input case is at most 300.
- **Subtask 3** [**20 points**]: The total length of the strings in an input case is at most 15 000.
- **Subtask 4** [**60 points**]: No additional constraint.

Examples

stdin	stdout
3 cafj aaaaaaa aabyyxll	4 2 5 2 3 2 8 8 8 8 8 8 21 10 9 1 8 1

Explanation

In the **first test case** the splits are:

1. $A =$ `""`; $B =$ `""`; $C =$ `"cafj"`.
2. $A =$ `""`; $B =$ `"c"`; $C =$ `"afj"`.
3. $A =$ `""`; $B =$ `"ca"`; $C =$ `"fj"`.
4. $A =$ `""`; $B =$ `"caf"`; $C =$ `"j"`.
5. $A =$ `""`; $B =$ `"cafj"`; $C =$ `""`.
6. $A =$ `"c"`; $B =$ `""`; $C =$ `"afj"`.
7. $A =$ `"c"`; $B =$ `"a"`; $C =$ `"fj"`.
8. $A =$ `"c"`; $B =$ `"af"`; $C =$ `"j"`.
9. $A =$ `"c"`; $B =$ `"afj"`; $C =$ `""`.
10. $A =$ `"ca"`; $B =$ `""`; $C =$ `"fj"`.
11. $A =$ `"ca"`; $B =$ `"f"`; $C =$ `"j"`.
12. $A =$ `"ca"`; $B =$ `"fj"`; $C =$ `""`.
13. $A =$ `"caf"`; $B =$ `""`; $C =$ `"j"`.
14. $A =$ `"caf"`; $B =$ `"j"`; $C =$ `""`.
15. $A =$ `"cafj"`; $B =$ `""`; $C =$ `""`.

Of these the following count for each scenario:

- Scenario `abc`: splits 1, 3, 4 and 11.
- Scenario `acb`: splits 2 and 5.
- Scenario `bac`: splits 1, 7, 8, 10 and 13.
- Scenario `bca`: splits 6 and 15.
- Scenario `cab`: splits 5, 12 and 14.
- Scenario `cba`: splits 9 and 15.