I split U in 3 (abc)

Valerio hat gerade T Strings $U_0, ..., U_{T-1}$ gefunden, wobei der i-te String aus N_i lateinischen Kleinbuchstaben besteht.

Da Valerio sehr neugierig ist, fragt er dich, für jedes $0 \leq i < T$, auf wie viele Arten man U_i in 3 möglicherweise leere Strings A, B, C aufteilen kann, sodass $U_i = A + B + C$ ist, unter den Einschränkungen jedes der folgenden Szenarien:

- Szenario `abc`: Die Aufteilungen müssen $A \preceq B \preceq C$ erfüllen;
- Szenario `acb`: Die Aufteilungen müssen $A \preceq C \preceq B$ erfüllen;
- Szenario `bac`: Die Aufteilungen müssen $B \preceq A \preceq C$ erfüllen;
- Szenario `bca`: Die Aufteilungen müssen $B \preceq C \preceq A$ erfüllen;
- Szenario `cab`: Die Aufteilungen müssen $C \preceq A \preceq B$ erfüllen;
- Szenario `cba`: Die Aufteilungen müssen $C \preceq B \preceq A$ erfüllen.

Dabei bezeichnet $+$ die String-Verkettung und $\preceq$ ist das lexikographische Kleiner-oder-Gleich.[1]

Implementierung

Du musst eine einzelne Datei mit der Endung `.cpp` einreichen.

⇦ Unter den Anhängen zu dieser Aufgabe findest du eine Vorlage `abc.cpp` mit einer Beispielimplementierung.

⚠ Eine einzelne Eingabedatei kann mehrere Testfälle enthalten! Stelle sicher, dass globale Variablen zwischen verschiedenen Läufen zurückgesetzt werden.

Du musst die folgende Funktion implementieren:

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

- Der Integer N repräsentiert die Länge des Strings U.
- Der String U ist einer der Strings, die Valerio gefunden hat.
- Die Funktion soll jedes Szenario beantworten, indem sie Werte den entsprechenden Parametern zuweist.
- Diese Funktion wird T Mal während der Ausführung deines Programms aufgerufen.

Der Grader wird die Funktionen aufrufen und die zurückgegebenen Werte in die Ausgabedatei schreiben.

[1] Formal gilt für zwei Strings S und T, dass $S \preceq T$ genau dann, wenn eine der folgenden Bedingungen wahr ist:
- S ist der leere String;
- Keiner der Strings ist leer, und das erste Zeichen von S kommt im lateinischen Alphabet vor dem ersten Zeichen von T.
- Keiner der Strings ist leer, die ersten Zeichen der beiden Strings sind gleich und $S' \preceq T'$, wobei S' und T' die Strings sind, die man erhält, indem man das erste Zeichen von S bzw. T entfernt.

Beispielgrader

Eine vereinfachte Version des Graders, der bei der Korrektur verwendet wird, ist im Verzeichnis zu diesem Problem verfügbar. Du kannst sie verwenden, um deine Lösungen lokal zu testen. Der Beispiel-Grader liest Eingabedaten aus `stdin`, ruft die von dir zu implementierende Funktion auf und schreibt in `stdout` im folgenden Format.

Die Eingabedatei besteht aus $T + 1$ Zeilen, wobei T die Anzahl der Testfälle ist, die Folgendes enthalten:

- Zeile 1: ein Integer T.
- Zeile $2 + i$ $(0 \leq i < T)$: ein String U_i.

Die Ausgabedatei besteht aus T Zeilen, die Folgendes enthalten:

- Zeile $1 + i$ $(0 \leq i < T)$: die 6 Antworten, die dein Programm für den i-ten Testfall gegeben hat, in derselben Reihenfolge, wie sie in der Aufgabenstellung präsentiert werden.

Einschränkungen

- Die Gesamtlänge der Strings in einem Eingabefall beträgt höchstens 400 000.
- Jeder der Strings ist nicht leer und besteht aus lateinischen Kleinbuchstaben.

Punktevergabe

Dein Programm wird auf mehreren Testfällen getestet, die in Unteraufgaben gruppiert sind. Die Punktzahl für eine Unteraufgabe entspricht der schlechtesten Punktzahl, die in einem ihrer Testfälle erzielt wurde, multipliziert mit dem Wert der Unteraufgabe.

Die Punktzahl für einen Testfall hängt davon ab, wie viele der sechs Szenarien du korrekt löst, gemäß der folgenden Tabelle:

Gelöste Szenarien	0	1	2	3	4	5	6
Punkte	0	0.3	0.5	0.7	0.8	0.9	1

- **Teilaufgabe 0** [**0 Punkte**]: Beispiel-Testfälle.
- **Teilaufgabe 1** [**10 Punkte**]: Das einzige Zeichen im String ist `a`.
- **Teilaufgabe 2** [**10 Punkte**]: Die Gesamtlänge der Strings in einem Eingabefall beträgt höchstens 300.
- **Teilaufgabe 3** [**20 Punkte**]: Die Gesamtlänge der Strings in einem Eingabefall beträgt höchstens 15 000.
- **Teilaufgabe 4** [**60 Punkte**]: Keine zusätzlichen Einschränkungen.

Beispiele

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

Erklärung

Im **ersten Testfall** sind die Aufteilungen:

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"`; $B =$ `""`; $C =$ `""`.

Davon zählen die folgenden für jedes Szenario:

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