I split U in 3 (abc)

Valerio ha appena trovato T stringhe $U_0, ..., U_{T-1}$. La i-esima stringa è composta da N_i lettere minuscole.

Dato che Valerio è molto curioso ti chiede, per ogni $0 \leq i < T$, in quanti modi si può partizionare U_i in 3 stringhe (possibilmente vuote) A, B, C tali che $U_i = A + B + C$ rispettando i vincoli di ciascuno dei seguenti scenari:

- Scenario `abc`: la partizione deve soddisfare $A \preceq B \preceq C$;
- Scenario `acb`: la partizione deve soddisfare $A \preceq C \preceq B$;
- Scenario `bac`: la partizione deve soddisfare $B \preceq A \preceq C$;
- Scenario `bca`: la partizione deve soddisfare $B \preceq C \preceq A$;
- Scenario `cab`: la partizione deve soddisfare $C \preceq A \preceq B$;
- Scenario `cba`: la partizione deve soddisfare $C \preceq B \preceq A$.

dove $+$ indica la concatenazione di stringhe e $\preceq$ è il minore o uguale lessicografico.[1]

Implementazione

Devi inviare un singolo file con estensione `.cpp`.

⇦ Tra gli allegati di questo problema, troverai un template `abc.cpp` con un'implementazione d'esempio.

⚠ Un singolo file di input potrebbe contenere più casi di test! Assicurati di resettare le variabili globali tra diverse esecuzioni.

Devi implementare la seguente funzione:

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);
```

- L'intero N rappresenta la lunghezza della stringa U.
- La stringa U è una delle stringhe che Valerio ha trovato.
- La funzione deve rispondere a ciascuno scenario assegnando i valori al parametro corrispondente.
- Questa funzione viene chiamata T volte durante l'esecuzione del tuo programma.

Il grader chiamerà la funzione e stamperà i valori restituiti nel file di output.

[1] Formalmente, date due stringhe S e T, si ha $S \preceq T$ se e solo se una delle seguenti condizioni è vera:
- S è la stringa vuota;
- Nessuna delle due stringhe è vuota, e il primo carattere di S precede il primo carattere di T nell'alfabeto latino.
- Nessuna delle due stringhe è vuota, i primi caratteri delle due stringhe sono uguali e $S' \preceq T'$ dove S' e T' sono le stringhe ottenute rimuovendo il primo carattere da S e T rispettivamente.

Grader di prova

Una versione semplificata del grader usato per la correzione è disponibile nella directory relativa a questo problema. Puoi usarla per testare le tue soluzioni in locale. Il grader legge i dati di input da `stdin`, chiama la funzione che devi implementare e scrive su `stdout` nel seguente formato.

Il file di input è composto da $T + 1$ righe, dove T è il numero di casi di test, contenenti:

- Riga 1: l'intero T.
- Riga $2 + i$ $(0 \leq i < T)$: la stringa U_i.

Il file di output è composto da T righe, contenenti:

- Riga $1 + i$ $(0 \leq i < T)$: le 6 risposte che il tuo programma ha dato per l'i-esimo caso di test, nello stesso ordine in cui sono presentate nel testo.

Assunzioni

- La lunghezza totale delle stringhe in un caso di input è al massimo 400 000.
- Ciascuna delle stringhe non è vuota ed è composta da lettere minuscole.

Assegnazione del punteggio

Il tuo programma sarà testato su diversi casi di test raggruppati in subtask. Il punteggio relativo a un subtask è uguale al punteggio peggiore ottenuto su uno dei suoi casi di test, moltiplicato per il valore del subtask.

Il punteggio per un caso di test dipende da quanti dei sei scenari risolvi correttamente, secondo la seguente tabella:

Scenari risolti	0	1	2	3	4	5	6
Punti	0	0.3	0.5	0.7	0.8	0.9	1

- **Subtask 0** [**0 punti**]: Casi d'esempio.
- **Subtask 1** [**10 punti**]: Ciascuna stringa è composta solo dal carattere `a`.
- **Subtask 2** [**10 punti**]: La lunghezza totale delle stringhe in un caso di input è al massimo 300.
- **Subtask 3** [**20 punti**]: La lunghezza totale delle stringhe in un caso di input è al massimo 15 000.
- **Subtask 4** [**60 punti**]: Nessuna limitazione aggiuntiva.

Esempi di input/output

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

Spiegazione

Nel **primo caso di test** le partizioni possibili sono:

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

A queste partizioni corrispondono i seguenti scenari:

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