Episodio IV: tracce di DNA (`dna`)

Terminata la gara e chiuso il buffet, lo staff felice torna nella sala di controllo per organizzare la premiazione. Ma una brutta sorpresa li attende: qualcuno ha cercato di infiltrarsi nel loro database per alterare i risultati! Anche se il tentativo si è infranto contro le impenetrabili misure di sicurezza dello staff, il colpevole non deve rimanere impunito del tentato crimine.

Un'attenta analisi rileva campioni organici sulla tastiera del server. Un primo tentativo di sequenziamento classico del DNA non va a buon fine: si deve trattare del particolare DNA degli alieni di Proxima B! I sospetti ricadono subito su ⏢⍀⟟⋔⍒⟟⏃⏁⟟, che già sembrava aver tentato di corrompere lo staff con una quantità sospetta di souvenir... ma non lo si può accusare senza una prova.

Figura 1: La procedura sperimentale di test del DNA alieno.

Il DNA alieno, differentemente da quello terrestre, ha solamente due tipi di nucleotidi chiamati `0` e `1`. Il suo studio è agli albori, e quindi non si può sequenziare direttamente il DNA di questo tipo. Tuttavia un test è stato recentemente sviluppato: data una sequenza di `0` e `1`, il test può stabilire se essa compare o meno (come sottostringa contigua) all'interno del campione di DNA.

Essendo queste analisi estremamente costose, lo staff vorrebbe sequenziare il campione di DNA facendo meno test possibile: aiutali ad organizzarsi al meglio!

Implementazione

Dovrai sottoporre un unico file, con estensione `.cpp`.

> ☞ Tra gli allegati a questo task troverai un template `dna.cpp` con un esempio di implementazione.

Dovrai implementare la seguente funzione:

C++
```
string analizza(int N);
```

- L'intero N rappresenta la lunghezza del campione di DNA alieno trovato.

- La funzione dovrà restituire una stringa di lunghezza N composta da soli caratteri `0` e `1`, contenente il sequenziamento del DNA campione.

Il tuo programma potrà utilizzare la seguente funzione, definita nel grader:

C++
```
bool test(string T);
```

- La stringa T deve avere lunghezza compresa tra 1 e N (inclusi) e deve essere composta dai soli caratteri `0` e `1`.
- La funzione restituisce `true` se la stringa T appare come sottostringa contigua del campione, e `false` altrimenti.

Il grader chiamerà la funzione `analizza` e ne stamperà il valore restituito sul file di output.

Grader di prova

Nella directory relativa a questo problema è presente una versione semplificata del grader usato durante la correzione, che potete usare per testare le vostre soluzioni in locale. Il grader di esempio legge i dati da `stdin`, chiama le funzioni che dovete implementare e scrive su `stdout`, secondo il seguente formato.

Il file di input è composto da due righe, contenenti:

- Riga 1: l'intero N.
- Riga 2: la stringa che descrive il campione.

Il file di output è composto da $Q + 1$ righe, dove Q è il numero di chiamate alla funzione `test` effettuate dal tuo programma, contenenti:

- Riga $1 + i$ ($0 \leq i < Q$): la stringa T passata alla i-esima chiamata a `test`.
- Riga $1 + Q$: la stringa restituita dalla funzione `analizza`.

Assunzioni

- $1 \leq N \leq 10\,000$.
- La funzione `test` può essere chiamata al massimo 30 000 volte.
- Diversamente dal grader di esempio, quello ufficiale è *adattivo*: la sequenza campione non è predeterminata ma viene decisa man mano che avvengono le chiamate a `test`.

Assegnazione del punteggio

Il tuo programma verrà testato su diversi casi di test. I casi di esempio sono inclusi nella correzione ma non vengono considerati per l'assegnazione del punteggio.

Ad un caso di test viene assegnato un punteggio di 0 se eccede il tempo limite o se la stringa restituita è errata. Altrimenti, sia Q il numero di query effettuate. Il punteggio dunque assegnato al caso di test è:

Q	Punteggio
$> 30\,000$	0
30 000	10
20 000	20
15 000	40
11 000	60
10 200	70
10 035	85
10 025	90
$\leq 10\,015$	100

Se Q è compreso fra due dei valori sopra il punteggio sarà intermedio, assegnato per interpolazione lineare. Il punteggio assegnato al problema sarà quindi il *peggior* punteggio ottenuto su un caso di test.

Esempi di input/output

stdin	stdout
`101010`	`Risposta corretta!` `5 chiamate a test.`

Puoi trovare altri esempi di input nella directory relativa a questo problema.

Spiegazione

Una possibile sequenza di chiamate nel caso di esempio può essere:

- `test("0")` restituisce `true`;
- `test("00")` restituisce `false`;
- `test("1")` restituisce `true`;
- `test("11")` restituisce `false`;
- a questo punto non ci possono essere due caratteri uguali consecutivi quindi il DNA deve essere `"010101"` oppure `"101010"`;
- `test("010101")` restituisce `false` quindi la risposta è `"101010"`.