Trova la permutazione (`permutazione2`)

Dario, Ermanno e Fabrizio hanno vinto le ICPC World Finals risolvendo tutti i problemi con due ore di anticipo. Dato che non possono uscire dalla sala fino alla fine della gara, decidono di giocare a "Trova la permutazione" sfidando Davide, Taulant e Tommaso.

Figura 1: L'icona di "Trova la permutazione"

Il gioco consiste nel trovare una permutazione $P_0, P_1, \ldots, P_{N-1}$ degli interi da 0 a $N-1$. La permutazione è nascosta, e si possono ottenere informazioni su di essa solo utilizzando uno scanner. In ogni momento, lo scanner si trova in una specifica posizione x ($0 \leq x \leq N-1$). Se lo scanner è in posizione x, esso controlla il prefisso $[0, x]$ della permutazione. Inoltre, lo scanner può avere due stati: `L` o `R`.

Inizialmente, la posizione è $x = 0$ e lo stato è `R`. Dario, Ermanno e Fabrizio possono effettuare 3 tipi di azioni:

1. Dario può scegliere un intero K e chiedere se esiste K nel prefisso $[0, x]$ della permutazione (x è la posizione attuale dello scanner).
2. Ermanno può cambiare lo stato dello scanner: se inizialmente lo stato era `L`, esso diventa `R` (e viceversa).
3. Fabrizio può spostare lo scanner. In particolare, se lo stato è `L`, x diminuisce di 1; se lo stato è `R`, x aumenta di 1. Dopo ogni azione, x deve essere compreso tra 0 e $N-1$.

Davide, Taulant e Tommaso hanno già giocato a "Trova la permutazione", effettuando rispettivamente A, B, C azioni di tipo 1, 2, 3. La squadra di Dario, Ermanno e Fabrizio vince se riesce a trovare la permutazione e, per ogni tipo di azione, riesce a effettuare al massimo tante azioni quante ne hanno usate gli avversari. Aiuta la squadra vincitrice delle ICPC World Finals a vincere anche questa sfida!

Implementazione

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

> ☞ Tra gli allegati a questo task troverai un template `permutazione2.cpp` con un esempio di implementazione.

Dovrai implementare la seguente funzione:

C++	`void indovina(int N, int A, int B, int C, vector<int>& H);`

La funzione viene chiamata durante l'esecuzione del programma con i seguenti parametri:

- L'intero N rappresenta la lunghezza della permutazione da indovinare.
- L'intero A rappresenta il numero massimo di azioni di tipo 1.
- L'intero B rappresenta il numero massimo di azioni di tipo 2.
- L'intero C rappresenta il numero massimo di azioni di tipo 3.
- L'array H è indicizzato da 0 a $N-1$ e inizializzato a 0.
- Al termine della chiamata l'array H deve contenere la permutazione nascosta.

Il tuo programma potrà utilizzare le seguenti funzioni, definite nel grader:

C++	`bool chiedi(int K);`

- La funzione invia in input allo scanner il numero K.
- K deve essere compreso tra 0 e $N-1$, altrimenti il programma termina con il messaggio `Domanda non valida`.
- Potrai usare questa funzione al più A volte, altrimenti il programma termina con il messaggio `Troppe chiamate (1)`.
- La funzione ritorna `true` se K è contenuto nel prefisso $[0, x]$ di P, altrimenti ritorna `false` (x è la posizione attuale dello scanner).

C++	`void stato();`

- La funzione cambia lo stato dello scanner: se inizialmente lo stato era `L`, esso diventa `R` (e viceversa).
- Potrai usare questa funzione al più B volte, altrimenti il programma termina con il messaggio `Troppe chiamate (2)`.

C++	`void sposta();`

- La funzione cambia la posizione x dello scanner. In particolare, se lo stato è `L`, x diminuisce di 1; se lo stato è `R`, x aumenta di 1.
- Al termine della chiamata, la posizione x deve essere compresa tra 0 e $N-1$, altrimenti il programma termina con il messaggio `Posizione non valida`.
- Potrai usare questa funzione al più C volte, altrimenti il programma termina con il messaggio `Troppe chiamate (3)`.

Grader di prova

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

L'input è composto da 2 righe, contenenti:

- Riga 1: gli interi N, A, B, C.
- Riga 2: l'array P da indovinare.

L'output è composto da 2 righe:

- Riga 1: l'array H al termine dell'esecuzione della funzione `indovina`.
- Riga 2: `Risposta corretta: ([a], [b], [c]) chiamate eseguite` se l'array H coincide con l'array P, `Risposta errata: ([a], [b], [c]) chiamate eseguite` altrimenti. a, b, c sono rispettivamente il numero di azioni di tipo 1, 2, 3 effettuate.

Assunzioni

- Consulta la sezione "Assegnazione del punteggio".

Assegnazione del punteggio

Il tuo programma verrà testato su diversi test case raggruppati in subtask. Per ottenere il punteggio relativo ad un subtask, è necessario risolvere correttamente tutti i test relativi ad esso.

- **Subtask 1** [**0 punti**]: Casi d'esempio.
- **Subtask 2** [**19 punti**]: $N = 100$, $A = 10000$, $B = 10000$, $C = 10000$
- **Subtask 3** [**12 punti**]: $N = 100$, $A = 6000$, $B = 6000$, $C = 6000$
- **Subtask 4** [**12 punti**]: $N = 100$, $A = 3000$, $B = 3000$, $C = 3000$
- **Subtask 5** [**25 punti**]: $N = 100$, $A = 1000$, $B = 1000$, $C = 10000$
- **Subtask 6** [**16 punti**]: $N = 1000$, $A = 10000$, $B = 10$, $C = 10000$
- **Subtask 7** [**16 punti**]: $N = 1000$, $A = 40000$, $B = 1$, $C = 2000$

Esempi di input/output

stdin	stdout
9 1000 1000 1000 3 2 7 8 6 0 5 4 1	3 2 7 8 6 0 5 4 1 Risposta corretta: (2, 1, 3) 3 chiamate eseguite

Spiegazioni

Presentiamo qui di seguito una possibile interazione che risolve correttamente il **primo caso d'esempio**:

- Inizialmente, la posizione è $x = 0$ e lo stato è `R`.
- Fabrizio sposta lo scanner (funzione `sposta()`). Poiché lo stato è `R`, la posizione x aumenta di 1 e diventa 1.
- Fabrizio sposta di nuovo lo scanner (funzione `sposta()`). Poiché lo stato è `R`, la posizione x aumenta di 1 e diventa 2.
- Dario chiede se 5 esiste nel prefisso $[0, x]$ di P (funzione `chiedi(5)`), ricevendo risposta `false`.
- Ermanno cambia lo stato dello scanner (funzione `stato()`). Il nuovo stato è `L`.
- Fabrizio sposta lo scanner (funzione `sposta()`). Poiché lo stato è `L`, la posizione x diminuisce di 1 e diventa 1.
- Dario chiede se 3 esiste nel prefisso $[0, x]$ di P (funzione `chiedi(3)`), ricevendo risposta `true`.
- Dario, Ermanno e Fabrizio, anche se non hanno informazioni sufficienti per determinare univocamente la permutazione, decidono di provare a indovinare. Dato che la permutazione da loro inviata (che si trova nell'array H al termine dell'esecuzione della funzione) è uguale a quella nascosta, il testcase è stato risolto. Sono state eseguite rispettivamente 2, 1, 3 azioni di tipo 1, 2, 3.