Dadi truccati (`dadi`)

Limite di tempo: 1.0 secondi
Limite di memoria: 256 MiB


Questo contratto vale L. 40.000

CONTRATTO
PARCO della VITTORIA

Rendita	- Solo terreno L.	5.000
»	Con 1 Casa »	20.000
»	Con 2 Case »	60.000
»	Con 3 Case »	140.000
»	Con 4 Case »	170.000
»	Con Albergo »	200.000

Se un giocatore possiede tutti i terreni d'uno stesso Gruppo (colore), la rendita del solo terreno viene raddoppiata.

Costo	di ogni Casa L.	20.000
»	di un Albergo più 4 Case »	20.000
Valore ipotecario . .	L.	20.000

L'ultima partita a Monopoly tra Gabriele e Giorgio è finita in tragedia dopo che Gabriele è stato costretto a ipotecare metà dei suoi terreni per coprire il costo dell'albergo di Giorgio in Parco della Vittoria.

Per evitare che la spiacevole situazione si ripeta un'altra volta, Gabriele sta mettendo in pratica gli insegnamenti del corso di ingegneria per costruire dei dadi robotici truccati. Il piano è molto astuto: i dadi sono progettati per fare in modo che Giorgio finisca sempre sulla casella più sfortunata, e che Gabriele si salvi sempre e paghi il minimo possibile ad ogni turno. Per raggiungere l'obiettivo, i dadi hanno all'interno un minuscolo giroscopio che permette loro di ruotare in modo da mostrare la faccia che decide Gabriele.

Gabriele non è uno sprovveduto, e sa che per non essere scoperto è necessario che nel lancio i dadi si muovano in modo realistico, compiendo molte rotazioni, all'apparenza casuali, ma che in realtà sono dettate dal software.

In questo momento Gabriele sta debuggando proprio questa parte di codice, verificando che i dadi compiano correttamente la sequenza di rotazioni imposta dal software. All'inizio della sequenza il dado si trova sempre nella posizione indicata in figura:

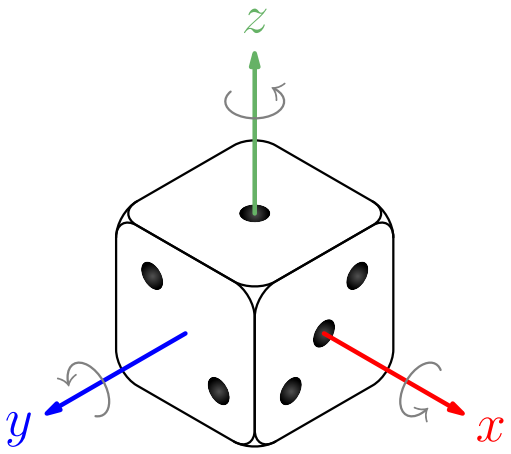


Figura 1. Situazione iniziale del dado.

Il driver che si occupa della rotazione accetta in input tre comandi: `X`, `Y`, e `Z`. Questi comandi hanno l'effetto di far compiere al dado una rotazione di 90 gradi rispetto all'asse scelto, in senso antiorario come mostrato dalle frecce grigie. Qui sotto sono rappresentati gli effetti dei comandi `X`, `Y`, e `Z` sul dado di Figura 1.

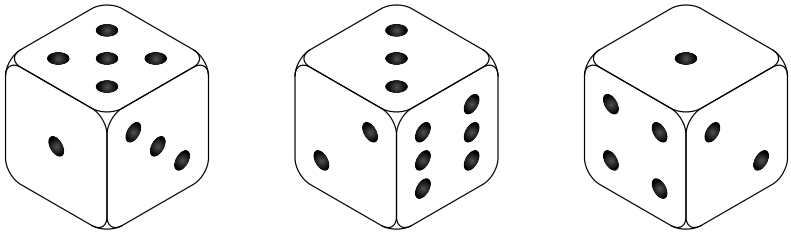


Figura 2. Effetti dei comandi `X`, `Y`, e `Z` sul dado di Figura 1.

Il driver supporta inoltre 3 comandi di interrogazione:

- `T`: ritorna il valore sulla faccia superiore del dado, ovvero quella corrispondente al verso positivo dell'asse z;
- `F`: ritorna il valore sulla faccia frontale del dado, ovvero quella corrispondente al verso positivo dell'asse y;
- `R`: ritorna il valore sulla faccia destra del dado, ovvero quella corrispondente al verso positivo dell'asse x.

Aiuta Gabriele a debuggare i dadi truccati, scrivendo un software di simulazione che, ricevuta la sequenza di comandi inviati al dado, calcoli quali sono le risposte corrette alle varie richieste `T`, `F` e `R` inoltrate al driver.

Implementazione

Dovrai sottoporre esattamente un file con estensione `.c`, `.cpp` o `.pas`.

> ☞ Tra gli allegati a questo task troverai un template (`dadi.c`, `dadi.cpp`, `dadi.pas`) con un esempio di implementazione da completare.

Se sceglierai di utilizzare il template, dovrai implementare la seguente funzione:

C/C++	`void simula(int N, char C[], int R[]);`
Pascal	`procedure simula(N: longint; var C: array of char; var R: array of longint);`

In cui:

- L'intero N rappresenta il numero di comandi inoltrati al driver.
- L'array `C`, indicizzato da 0 a $N-1$, contiene i comandi (un carattere tra `X, Y, Z, T, F, R`).
- La funzione dovrà riempire l'array di interi `R`, indicizzato da 0 a $M-1$, dove M rappresenza il numero di comandi di interrogazione (ovvero `T, F, R`). Nella posizione i scrivere la risposta all'i-esimo comando di interrogazione.

Dati di input

Il file `input.txt` è composto da due righe. La prima riga contiene l'unico intero N. La seconda riga contiene gli N caratteri C_i, senza spazi.

Dati di output

Il file `output.txt` è composto da un'unica riga contenente gli M interi di risposta.

Assunzioni

- $1 \leq N \leq 100\,000$.
- I comandi C_i sono sempre un carattere tra `X`, `Y`, `Z`, `T`, `F` e `R`.

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** [**10 punti**]: Casi d'esempio.
- **Subtask 2** [**20 punti**]: $N \leq 100$.
- **Subtask 3** [**40 punti**]: $N \leq 1000$.
- **Subtask 4** [**30 punti**]: Nessuna limitazione specifica.

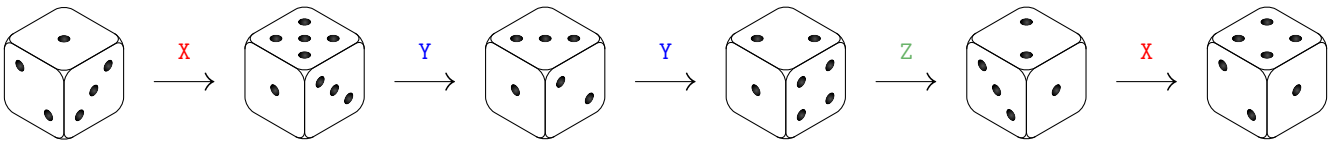
Esempi di input/output

input.txt	output.txt
9 XYYTZFXFR	2 3 2 1

input.txt	output.txt
10 TZZYXFRTXR	1 4 6 2 6

Spiegazione

Nel **primo caso di esempio** il dado subisce queste rotazioni:



Nel **secondo caso di esempio** il dado subisce queste rotazioni: