Il gambero scrivano

Alcune persone dicono che Leonardo fosse un grande ammiratore di Johannes Gutenberg, il fabbro tedesco che ha inventato la stampa a caratteri mobili, e che per omaggiarlo abbia inventato una macchina chiamata *il gambero scrivano* (crayfish scrivener) — una macchina da scrivere molto semplice. Il gambero scrivano è abbastanza simile a una moderna macchina da scrivere e accetta solo due comandi: uno per scrivere il carattere successivo ed uno per annullare i comandi più recenti. La peculiarità di questa macchina è che il comando di annullamento è estremamente potente: un annullamento è esso stesso un comando, e come tale può essere a sua volta annullato.

Descrizione del problema

Il tuo compito è quello di realizzare una versione software del gambero scrivano: inizia con un testo vuoto e accetta una sequenza di comandi inseriti dall'utente, e query relative a specifiche posizioni della corrente versione del testo, nel modo seguente:

- `Init`() — chiamata una volta sola all'inizio dell'esecuzione, senza argomenti. Può essere usato per inizializzare le strutture dati. Non sarà mai annullato.
- `TypeLetter`(L) — aggiunge alla fine del testo un singolo carattere minuscolo L scelto tra `a`, ..., `z`.
- `UndoCommands`(U) — annulla gli ultimi U comandi, dove U è un intero positivo.
- `GetLetter`(P) — restituisce la lettera nella posizione P del testo corrente, dove P è un intero non negativo. La prima lettera nel testo ha indice 0. (Questa query non è un comando e quindi è ignorata dal comando di annullamento.)

Dopo la chiamata iniziale a `Init`(), le altre funzioni possono essere chiamate zero o più volte, in qualsiasi ordine. È garantito che il valore di U non supererà il numero di comandi precedentemente ricevuti, e che P sarà inferiore alla lunghezza del testo corrente (il numero di lettere del testo corrente).

Per quanto riguarda `UndoCommands`(U), esso annulla i precedenti U comandi in ordine *inverso*: se il comando da annullare è `TypeLetter`(L), allora rimuove L dalla fine del testo corrente; se il comando da annullare è `UndoCommands`(X) per qualche valore X, esso ri-esegue gli X comandi precedenti nel loro ordine *originale*.

Esempio

Mostriamo una possibile sequenza di chiamate, insieme allo stato del testo dopo ogni chiamata.

Chiamata	Risultato	Testo corrente
Init()		
TypeLetter(a)		a
TypeLetter(b)		ab
GetLetter(1)	b	ab
TypeLetter(d)		abd
UndoCommands(2)		a
UndoCommands(1)		abd
GetLetter(2)	d	abd
TypeLetter(e)		abde
UndoCommands(1)		abd
UndoCommands(5)		ab
TypeLetter(c)		abc
GetLetter(2)	c	abc
UndoCommands(2)		abd
GetLetter(2)	d	abd

Subtask 1 [5 punti]

Il numero complessivo di comandi e query è compreso fra 1 e 100 (estremi inclusi) e non vi saranno chiamate a `UndoCommands`.

Subtask 2 [7 punti]

Il numero complessivo di comandi e query è compreso fra 1 e 100 (estremi inclusi) e nessun `UndoCommands` sarà annullato.

Subtask 3 [22 punti]

Il numero complessivo di comandi e query è compreso fra 1 e 5 000 (estremi inclusi).

Subtask 4 [26 punti]

Il numero complessivo di comandi e query è compreso fra 1 e 1 000 000 (estremi inclusi). Tutte le chiamate a `GetLetter` saranno successive a tutte le chiamate a `TypeLetter` e `UndoCommands`.

Subtask 5 [40 punti]

Il numero complessivo di comandi e query è compreso fra 1 e 1 000 000 (estremi inclusi).

Dettagli implementativi

Devi inviare esattamente un file, chiamato `scrivener.c`, `scrivener.cpp` oppure `scrivener.pas`. Questo file deve implementare le funzioni sopra descritte utilizzando i seguenti prototipi.

Programmi C/C++

```
void Init();
void TypeLetter(char L);
void UndoCommands(int U);
char GetLetter(int P);
```

Programmi Pascal

```
procedure Init;
procedure TypeLetter(L : Char);
procedure UndoCommands(U : LongInt);
function GetLetter(P : LongInt) : Char;
```

Queste funzioni devono comportarsi come descritto sopra. Ovviamente sei libero di implementare altre funzioni per uso interno. Le tue sottoposizioni non devono interagire in nessun modo con l'input/output standard né con nessun altro file.

Grader di esempio

Il grader di esempio legge l'input nel seguente formato:

- linea 1: il numero complessivo di comandi e query nell'input;
- su ciascuna linea successiva:
  - `T` seguito da uno spazio e una lettera minuscola per un comando `TypeLetter`;
  - `U` seguito da uno spazio e un numero intero per `UndoCommands`;
  - `P` seguito da uno spazio e un numero intero per `GetLetter`.

Il grader di esempio stamperà i caratteri restituiti da `GetLetter`, ciascuno su una riga separata.