Cangrejo Escribano

Cuenta la leyenda que Leonardo era un gran admirador de Johannes Gutenberg, un herrero aleman que inventó la imprenta, y al cual rindió homenaje diseñando una máquina llamada el cangrejo escribano. Es similar a una simple máquina de escribir, pero acepta sólo dos comandos: uno para escribir el siguiente carácter, y uno para deshacer los comandos más recientes. La característica más notable del cangrejo escribano es que el comando para deshacer es extremadamente potente: cada deshacer también se considera un comando en sí mismo, y puede ser deshecho.

Enunciado

Tu tarea es crear un programa para el cangrejo escribano: empieza con un texto vacío y acepta una secuencia de comandos que entra el usuario, y consultas sobre posiciones específicas de la versión actual del texto, como sigue.

- `Init`() — esta función es llamada una sola vez al principio del programa, sin argumentos. Puede servir para inicializar estructuras de datos. Nunca se pedirá que se deshaga.
- `TypeLetter`(L) — añade al final del texto una sola letra minúscula L entre `a`, …, `z`.
- `UndoCommands`(U) — deshace los últimos U comandos, siendo U un entero positivo.
- `GetLetter`(P) — devuelve la letra en la posición P en el texto actual para un índice no negativo P. La primera letra del texto tiene índice 0. (Esta consulta no es un comando y por lo tanto no se podrá deshacer con un comando para deshacer).

Después de la llamada inicial a `Init` (), las otras rutinas pueden ser llamadas cero o más veces en cualquier orden. Se te garantiza que U no excederá el número de comandos recibidos previamente, y que P será menor que la longitud actual del texto (número de letras del texto actual).

Respecto a `UndoCommands`(U), deshace los U comandos previos en orden *inverso*: si el comando a deshacer es `TypeLetter`(L), entonces elimina L del final del texto actual; si el comando a deshacer es `UndoCommands`(x) para algun valor x, entonces rehace los anteriores x comandos en el orden *original*.

Ejemplo

Mostramos una posible secuencia de llamadas, junto al estado del texto después de cada llamada.

Llamada	Retorna	Texto actual
`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 points]

- El número total de comandos y consultas está entre 1 y 100 (incluidos) y no habrá llamadas a `UndoCommands`.

Subtask 2 [7 points]

- El número total de comandos y consultas está entre 1 y 100 (incluidos) y no se deshará ningun `UndoCommands` .

Subtask 3 [22 points]

- El número total de comandos y consultas está entre 1 y 5 000 (incluidos).

Subtask 4 [26 points]

- El número total de comandos y consultas está entre 1 y 1 000 000 (incluidos). Todas las llamadas a `GetLetter` tendrán lugar después de todas las llamadas a `TypeLetter` y `UndoCommands`.

Subtask 5 [40 points]

- El número total de comandos y consultas está entre 1 y 1 000 000 (incluidos).

Detalles de implementación

Tienes que enviar exactamente un fichero, llamado `scrivener.c`, `scrivener.cpp` o `scrivener.pas`. Este fichero debe implementar los subprogramas descritos anteriormente usando las siguientes cabeceras.

Programas en C/C++

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

Programas en Pascal

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

Estos subprogramas se deben comportar como se ha descrito anteriormente. Por supuesto pueden usar otros subprogramas que tu implementes. Tus envíos no deben interactuar de ninguna manera con la entrada y salida estándar, ni con otros ficheros.

Sample grader

El sample grader lee la entrada en el formato siguiente:

- línea 1: el número total de comandos y de consultas en la entrada;
- en cada línea siguiente:
  - `T` seguido de un espacio y una letra minúscula para un comando `TypeLetter`;
  - `U` seguido de un espacio y un entero para un comando `UndoCommands`;
  - `P` seguido de un espacio y un entero para un comando `GetLetter`.

El sample grader escribirá los caracteres devueltos por `GetLetter`, cada uno en una línea separada.