Cangrejo Escribano

Algunas personas dicen que Leonardo fue un gran admirador de Johannes Gutenberg, el herrero alemán que inventó la impresión de tipos móviles y que como un homenaje diseñó una maquina llamada – "il gambero scrivano" – el cual era un dispositivo de escritura bastante simple. Dicho dispositivo era similar a una maquina de escribir moderna y aceptaba dos comandos: uno para escribir el siguiente carácter y otro para deshacer los comandos más recientes. La característica más notable del cangrejo escribano es qué su comando para deshacer es extremadamente poderoso dado que al ser él mismo considerado como un comando, puede ser deshecho.

Problema

Tu tarea consiste en implementar una versión en software del cangrejo escribano: debe iniciar con un texto vacío, aceptar una secuencia de comandos ingresados por el usuario y preguntas para posiciones especificas sobre la versión actual del texto, como se muestra a continuación.

- `Init`() – llamada una vez al inicio de la ejecución, sin argumentos. Puede ser usada para inicializar estructuras de datos y nunca deberá ser deshecha.
- `TypeLetter`(L) —agrega al final del texto la letra L seleccionada de `a`, …, `z`.
- `UndoCommands`(U) – deshace los últimos U comandos, para un entero positivo U.
- `GetLetter`(P) – regresa la letra en la posición P, para un índice no negativo P. La primera letra del texto tiene el índice 0. (Esta pregunta no es un comando y es por esto que es ignorada por el comando deshacer.)

Las funciones de arriba pueden ser llamadas cero o más veces en cualquier orden, se garantiza que U no excederá la cantidad de comandos previamente recibidos y P será menor que la longitud del texto en ese momento (la cantidad de letras en el texto).

Como `UndoCommands`(U), deshace los U previos comandos en "reversa": si el comando a ser deshecho es `TypeLetter`(L), entonces quita L del final del texto actual, en cambio si el comando a ser deshecho es `UndoCommands`(X), para algún valor de X, entonces rehace los X comandos previos en su orden "original".

"" Ejemplo ""

Mostramos una posible secuencia de llamadas junto con el estado del texto después de cada llamada.

Llamada	Regresa	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`

Sub-tarea 1 [5 puntos]

- La cantidad de comandos y preguntas estará entre 1 y 100 (inclusive) y no habrá llamadas a `UndoCommands`

Sub-tarea 2 [7 puntos]

- La cantidad de comandos y preguntas estará entre 1 y 100 (inclusive) y no habrá comandos `UndoCommands` que deban ser deshechos.

Sub-tarea 3 [22 puntos]

- La cantidad de comandos y preguntas estará entre 1 y 5,000 (inclusive).

Sub-tarea 4 [26 puntos]

- La cantidad de comandos y preguntas estará entre 1 y 1,000,000 (inclusive). Todas las llamadas a `GetLetter` ocurrirán después de todas las llamadas a `TypeLetter` y `UndoCommands`.

Sub-tarea 5 [40 puntos]

- La cantidad de comandos y preguntas estará entre 1 y 1,000,000 (inclusive).

Detalles de implementación

Tienes que enviar exactamente un archivo llamado `scrivener.c`, `scrivener.cpp` o `scrivener.pas`. Este archivo debe implementar las funciones descritas arriba usando las firmas siguientes.

“” 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;
```

Dichas funciones deberán comportarse de acuerdo a la descripción anterior. Está claro que eres libre de implementar funciones adicionales para uso interno. Tus envíos no deberán interactuar de ninguna manera con la entrada/salida estándar ni con algún archivo.

“” Evaluador de ejemplo (Grader) “”

El evaluador de ejemplo lee de la entrada estándar el siguiente formato:

- línea 1: el número de comandos en la entrada;
- en cada una de las siguientes líneas:
  - `T` seguido por un espacio y una letra en minúscula para un comando `TypeLetter`;
  - `U` seguido por un espacio y un entero para un comando `UndoCommands`;
  - `P` seguido por un espacio y un entero para `GetLetter`

El evaluador ejemplo imprimirá los caracteres devueltos por `GetLetter`, cada uno en una línea separada.