IOI '12 P3 - Crayfish Scrivener

Some people say that Leonardo was a great admirer of Johannes Gutenberg, the German blacksmith who invented movable-type printing, and that he paid homage by designing a machine called the crayfish scrivener — *il gambero scrivano* — a very simple typing device. It is somehow similar to a simple modern typewriter and accepts only two commands: one to type the next character and one to undo the most recent commands. The notable feature of the crayfish scrivener is that the undo command is extremely powerful: an undo is also considered to be a command itself, and can be undone.

Statement

Your task is to realize a software version of the crayfish scrivener: it starts with an empty text and accepts a sequence of commands entered by the user, and queries for specific positions of the current version of the text, as follows.

- `Init()` — called once at the beginning of the execution, without arguments. It can be used for initializing data structures. It will never need to be undone.
- `TypeLetter(L)` — append to the end of the text a single lowercase letter L chosen from $a, \dots, z$.
- `UndoCommands(U)` — undo the the last U commands, for a positive integer U.
- `GetLetter(P)` — return the letter at position P in the current text, for a non-negative index P. The first letter in the text has index 0. (This query is not a command and thus is ignored by the undo command.)

After the initial call to `Init()`, the other routines can be called zero or more times in any order. It is guaranteed that U will not exceed the number of previously received commands, and that P will be less than the current text length (the number of letters in the current text).

As for `UndoCommands(U)`, it undoes the previous U commands in reverse order: if the command to be undone is `TypeLetter(L)`, then it removes L from the end of the current text; if the command to be undone is `UndoCommands(X)` for some value X, it re-does the previous X commands in their original order.

Example

We show a possible sequence of calls, together with the state of the text after each call.

Call	Returns	Current text
`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]

- The total number of commands and queries is between 1 and 100 (inclusive) and there will be no calls to `UndoCommands`.

Subtask 2 [7 points]

- The total number of commands and queries is between 1 and 100 (inclusive) and no `UndoCommands` will be undone.

Subtask 3 [22 points]

- The total number of commands and queries is between 1 and $5\,000$ (inclusive).

Subtask 4 [26 points]

- The total number of commands and queries is between 1 and $1\,000\,000$ (inclusive). All calls to `GetLetter` will occur after all calls to `TypeLetter` and `UndoCommands`.

Subtask 5 [40 points]

- The total number of commands and queries is between 1 and $1\,000\,000$ (inclusive).

Implementation details

You must implement the subprograms described above using the following signatures.

C/C++ programs

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

Pascal programs

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

These subprograms must behave as described above. Of course you are free to implement other subprograms for their internal use. Your submissions must not interact in any way with standard input/output, nor with any other file.