Наборщик-рак

Некоторые люди говорят, что Леонардо был большим почитателем Иоганна Гутенберга, немецкого кузнеца, который изобрел подвижную (наборную) печать, и что он воздал должное, сконструировав машину, названную им наборщик-рак — *il gambero scrivano* — очень простое наборное устройство. Оно чем-то похоже на современную простою пишущую машинку и выполняет всего 2 команды: одна, чтобы напечатать следующий символ, и вторая, чтобы отменить несколько последних команд. Замечательным свойством наборщика-рака является исключительная мощность команды отмены, которая рассматривается сама по себе как команда и тоже может быть отменена.

Условие

Вам необходимо реализовать программную модель наборщика-рака: она начинает работу с пустого текста, обрабатывает последовательность команд, передаваемых ей пользователем, и запросы относительно определенных позиций в текущем состоянии текста, как описано ниже.

- `Init`() — вызывается один раз в начале выполнения, без аргументов. Может использоваться для инициализации структур данных. Эта операция никогда не отменяется.
- `TypeLetter`(L) — добавляет в конец текста один символ L - маленькую букву из диапазона `a`, …, `z`.
- `UndoCommands`(U) — отменяет последние U команд, где U - положительное целое число.
- `GetLetter`(P) — возвращает символ - букву, находящуюся в позиции P текущего текста, где P - неотрицательное целое число. Первая буква текста имеет индекс 0. (Этот запрос не является командой и поэтому игнорируется командой отмены.)

После начального вызова `Init`() другие процедуры могут вызываться ноль или более раз в любом порядке. Гарантируется, что U не будет превышать количество ранее полученных команд и что P будет меньше чем текущая длинна текста (количество букв в текущем тексте).

Вызов `UndoCommands`(U) отменяет предыдущие U команд в *обратном* порядке. Если отменяемая команда - это `TypeLetter`(L), то из конца текста удаляется буква L. Если отменяемая команда - это `UndoCommands`(X), то для этого значения X она заново применяет предыдущие X команд в их *оригинальном* порядке.

Пример

Ниже приведена последовательность вызовов вместе с состоянием текста после каждого из вызовов.

Вызов	Результат	Текущий текст
`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`

Подзадача 1 [5 баллов]

- Общее количество команд и запросов находится в диапазоне от 1 до 100 (включительно) и нет вызовов `UndoCommands`.

Подзадача 2 [7 баллов]

- Общее количество команд и запросов находится в диапазоне от 1 до 100 (включительно) и нет отмен команд `UndoCommands`.

Подзадача 3 [22 баллов]

- Общее количество команд и запросов находится в диапазоне от 1 до 5 000 (включительно).

Подзадача 4 [26 баллов]

- Общее количество команд и запросов находится в диапазоне от 1 до 1 000 000 (включительно). Все вызовы `GetLetter` происходят после всех вызовов `TypeLetter` и `UndoCommands`.

Подзадача 5 [40 баллов]

- Общее количество команд и запросов находится в диапазоне от 1 до 1 000 000 (включительно).

Детали реализации

Вы должны сдать только один файл с именем `scrivener.c`, `scrivener.cpp` или `scrivener.pas`. Эта программа должна реализовывать процедуры, описанные выше, с такими сигнатурами.

программы на C/C++

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

программы на Pascal

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

Эти подпрограммы должны вести себя как описано выше. Естественно вы имеете право реализовывать другие подпрограммы для внутреннего пользования. Ваши решения не должны взаимодействовать ни коим образом со стандартным вводом/выводом, и ни с каким другим файлом.

Пример проверяющего модуля

Предоставленный проверяющий модуль читает ввод в таком формате:

- строка 1: общее количество команд и запросов на входе;
- в каждой следующей строке:
  - `T`, за которым следует пробел и маленькая буква для команды `TypeLetter`;
  - `U`, за которым следует пробел и целое число для команды `UndoCommands`;
  - `P`, за которым следует пробел и целое число для запроса `GetLetter`.

Предоставленный проверяющий модуль печатает возвращаемые `GetLetter` буквы, по одной в строке.