NOI '03 P2 - Text Editor

Time limit: 0.25s **Memory limit:** 64M

National Olympiad in Informatics, China, 2003

A long, long time ago, DOS3.x programmers were starting to get tired of EDLIN. So they gradually switched to editors that they wrote themselves.

Many years later, Alex has accidentally stumbled upon a text editor from back in these days. After some basic testing, he has made a shocking discovery: this software is able to perform tens of thousands of edit operations per second (of course, you cannot conduct these tests by hand)! So, Alex is spending sleepless nights trying to create a product like this himself. Can you help him?

To clarify the objective, Alex has created some definitions to make the term "text editor" less abstract:

- **text**: a sequence formed using 0 or more ASCII characters with values in the inclusive range $[32, 126]$ (i.e. spaces and visible characters).
- **cursor**: a mark within a block of text to indicate a position. It may be positioned at the beginning of the text, the end of the text, or between any two adjacent characters in the text.
- **text editor**: a program containing a block of text and a single cursor, capable of performing the following six operations. If the block of text is empty, we will consider the text editor to be empty.

Operation	Input Format	Function
Move(k)	`Move k`	Moves the cursor's position to immediately after the k-th character in the text. If $k = 0$, then the cursor is moved to the start of the text.
Insert(n, s)	`Insert n↵ s`	Inserts a string s of length n ($n \geq 1$) at the position of the cursor. The position of the cursor does not change.
Delete(n)	`Delete n`	Delete the n ($n \geq 1$) characters following the cursor. The position of the cursor does not change.
Get(n)	`Get n`	Outputs the n ($n \geq 1$) characters following the cursor. The position of the cursor does not change.
Prev	`Prev`	Moves the cursor backwards by one character.
Next	`Next`	Moves the cursor forwards by one character.

For example, an empty text editor when given the operations: Insert(13, `Balanced tree`), Move(2), Delete(5), Next(), Insert(7, `editor`), Move(0), and Get(16), will output `Bad editor tree`.

Your task is to:

- Create an empty text editor.
- Read some operations from the input and execute them.
- For each Get() operation, output the correct contents.

Input Specification

The first line of input contains an integer t, the number of operations. The following lines contain t operations.
To make text from the text editor easier to read, the string following the Insert() operation may contain multiple line breaks. You must ignore them (if this is hard to understand, please refer to the sample input and output below).
Other than newline characters, all of the characters in the input will have ASCII values within the range $[32, 126]$, inclusive. There will be no trailing spaces.

You can assume that for each test case:

- The number of Move() operations will not exceed $50\,000$. The total number of Insert() and Delete() operations will not exceed 4000. The total number of Prev() and Next() operations will not exceed $200\,000$.
- The number of characters inserted in each Insert() operation will not exceed 2M (1M $= 1024 \times 1024$). The length of the correct output will not exceed 3M.
- For each Delete() and Get() operation, there will always be sufficient characters following the cursor. The Move(), Prev(), and Next() operations will never move the cursor to an invalid position.
- All operations will be valid.

Output Specification

Each line of the output should contain the text outputted by a single corresponding Get() operation.

Sample Input

```
15
Insert 26
abcdefghijklmnop
qrstuv wxy
Move 16
Delete 10
Move 5
Insert 1
^
Next
Insert 1
_
Next
Next
Insert 4
.\/.
Get 4
Prev
Insert 1
^
Move 0
Get 22
```

Sample Output

```
.\/.
abcde^_^f.\/.ghijklmno
```

Problem translated to English by **Alex**.