2008 ACM ICPC South Central USA Regional Programming Contest

Tribute (Editor)

Introduction:

This problem is not about the greatest editor in the world; it is just a tribute.

The *Tenacious eDitor* is an attempt to clone to power of a certain modal editor. However, it is notoriously bug-ridden. You've been asked to replace Kyle (who quit the devteam) and help with fixing some of the bugs.

The main problem with the current release of `td`, as it is commonly referred to, is that it cannot differentiate between text on a line and commands. Whenever a user wants to type an `x`, the editor interprets it as a command to delete the last character. This is problematic. Thankfully, commands are case-sensitive, and each letter is at most a single command (sometimes uppercase, sometimes lowercase), so one can manage to write any given line of text as long as it's allowed to look like something written by a fourteen-year-old hacker wannabe.

Editing occurs in an *edit buffer*, which starts out empty. The cursor in `td` rests in *gaps*, locations surrounding each printed character. When a non-command character is entered, the character is inserted to the right of the cursor's current location and the cursor moves to the gap to the right of the new character. (This is the standard behavior of most editors.) A line with no characters has a single gap. A line with a single printable character has two (one before and one after), a line with two printable characters has three, and so on.

The currently-implemented commands are:

Command keys	Result	Cursor location after operation
`x`	Delete the character immediately to the left of the cursor	One gap to the left
`K`	Delete all non-spaces to the left of the cursor, stopping at the first space encountered or the beginning of the line; if the previous character is a space, delete nothing	The gap to the right of the "stop" character
`c`	Duplicate the character to the left of the cursor, inserting it to the right of the cursor	The gap to the right of the duplicate character (one right)
`D`	Duplicate all non-spaces to the left of the cursor, stopping at the first space encountered or the beginning of the line and inserting them to the right of the cursor; if the previous character is a space, duplicate nothing	The gap to the right of the rightmost duplicate character
`R`	Reverse all non-spaces to the left of the cursor, stopping at the first space encountered or the beginning of the line; if the previous character is a space, reverse nothing	No change
`p`	Delete all characters to the left of the cursor	Leftmost gap
`W`	Delete all characters to the right of the cursor	No change
`h`	Move the cursor one gap to the left	One gap to the left
`L`	Move the cursor one gap to the right	One gap to the right
`f`	Move the cursor to the first gap on the line	Leftmost gap
`G`	Move the cursor to the last gap on the line	Rightmost gap
any non-command character	Insert the character into the edit buffer to the right of the cursor	One gap to the right

For the purposes of this problem, the only characters are uppercase and lowercase characters, digits, and spaces.

When a character is added internally to a line of text, all subsequent characters shift to the right (as in "insert mode"); when characters are deleted from anywhere other than the end of a line, all subsequent characters shift to the left.

Any commands which cannot properly do anything (attempting to move past the first or last gap, deleting characters when none exist, and so on) do nothing. They do **not** show up as printed characters.

Each line represents a different session of `td`; at the beginning of a new session, the edit buffer is empty.

Given a series of keystrokes by a user, can you determine what text actually results if it were entered in the current version of `td`?

Input:

Input to this problem will begin with a line containing a single integer N ($1 \le N \le 100$) indicating the number of data sets.

Each data set consists of a single line of characters, as defined above, with no leading or trailing whitespace. There are no less than 1 and no more than 100 characters per line. They represent the series of keystrokes entered by a user for a particular session of `td`.

Output:

For each data set, print the final state of the edit buffer from the sequence of characters entered, with a carat (`^`) at the location of the cursor.

Sample Input:

```
2
Deletex bigdeleteKduPc bigduPD esreverR midinhhdleGxx midelletehhhxGfmoo
THis ratHer obnoXious line oF CHaraCters surPrisingly triggers no Commands
```

Sample Output:

```
moo^elet duPP bigduPbigduP reverse middle midelete
THis ratHer obnoXious line oF CHaraCters surPrisingly triggers no Commands^
```