Programming 1 — Homework assignment 9

Deadline: Sunday, January 13, 2019, at 23:55

Editor

Task description

Write a program that acts as a simple text editor. The program should start with an empty document and then respond to the following commands:

- `#` *text*

  Adds a line with a given text at the end of the document.

- `+` *index* *text*

  Inserts a line with a given text before the line with a given index. The first line has index 0.

- `-` *index*

  Removes the line with the given index.

- `<`

  Undoes the most recently executed command.

- `>`

  Redoes the most recently undone command.

- `x`

  Quit the program.

Naturally, the editor having such a rich command set cannot be given away for free. The user is charged for every command except `x`. The command `#` costs z dinars, with z being the number of characters in the string *text*. The price of the command `+` is calculated using the formula $(2v+z)$, where v denotes the number of lines between the insertion point and the end of the document, and z represents the number of characters in the string *text*. The price of the command `-` amounts to $(3v + 2z)$ dinars, where v is the number of lines in the document that follow the line to be removed, and z is the number of characters in that line, without counting the final newline character.

The command `>` costs the same as the command that is being repeated. The command `<` costs as much as the operation that has to be executed to undo the corresponding command. For example, to undo a command that inserts a line, the inserted line has to be removed. The undo operation therefore costs as much as the removal of that line.

Your program should sequentially read and execute commands until arriving at the command `x`. After executing each command (except for the command `x`), your program should print the amount of money spent so far and the current contents of the document.

Input

The input consists of at most 100 lines. Each line contains a command and, if needed, parameters. The parameter *index* is an integer between 0 and n, inclusive, in the case of inserting a line, and an integer between 0 and $(n-1)$, inclusive, in the case of removing a line (n denotes the current number of lines). The parameter *text* is a string of length between 1 and 100, inclusive, consisting of uppercase and lowercase letters of the English alphabet, digits, and underscores.

A command `<` may occur anywhere except in the first line of the input and in the cases where we have undone all the commands executed so far. A command `>` may occur only immediately after a command `<` or `>`. A sequence of k commands `<` may be followed by at most k commands `>`. In all test cases, there is exactly one command `x`, and it occurs in the last line of the input.

Following are the properties of the individual test cases:

- J1–J2, S1–S10: contain only commands `#` and `x`.
- J3–J4, S11–S20: may also contain commands `+`.
- J5–J6, S21–S30: may also contain commands `-`.
- J7–J8, S31–S40: may also contain commands `<`. Test cases J7 and S31–S35 contain at most one command `<` in the entire input.
- J9–J10, S41–S50: may also contain command `>`.

Output

For each command except `x`, print a line of the form

moneySpent `|` *document*

where *moneySpent* denotes the total amount of money spent so far, and *document* denotes the current contents of the document, printed in the form $v_1/v_2/\ldots/v_n$, where v_i (for $i = 1, \ldots, n$) is the i-th line of the document.

Test case J9 ...

... is shown in Figure 1. The command `+ 0 Vrba`, for instance, costs $2 \cdot 2 + 4 = 8$ dinars, since there are $v = 2$ lines (`srecna` and `vas`) between the point of insertion and the end of the document, and the inserted line comprises $z = 4$ characters. The command `- 2` costs $3 \cdot 3 + 2 \cdot 6 = 21$ dinars: the line to be removed consists of $z = 6$ characters and is followed by $v = 3$ lines (`draga`, `vas`, and `domaca`).

Additional instructions

The commands and the parameters *text* should be read by calling `sc.next()`. The parameters *index* should be read by invoking `sc.nextInt()`. In both cases, `sc` is a `Scanner` object. The length of a string `niz` is obtained using `niz.length()`.

Can the classes that represent commands `#`, `+`, and `-` be arranged in a hierarchy? What is common to these three types of commands? Can a command `#` be regarded as a special case of a command `+`?

Input:

```
# srecna
+ 1 vas
+ 0 Vrba
+ 2 draga
# domaca
+ 0 0
- 2
<
<
<
>
- 3
<
<
<
<
>
>
>
<
<
<
<
<
x
```

Output:

```
6 | srecna
9 | srecna/vas
17 | Vrba/srecna/vas
24 | Vrba/srecna/draga/vas
30 | Vrba/srecna/draga/vas/domaca
41 | 0/Vrba/srecna/draga/vas/domaca
62 | 0/Vrba/draga/vas/domaca
74 | 0/Vrba/srecna/draga/vas/domaca
91 | Vrba/srecna/draga/vas/domaca
103 | Vrba/srecna/draga/vas
109 | Vrba/srecna/draga/vas/domaca
118 | Vrba/srecna/draga/domaca
123 | Vrba/srecna/draga/vas/domaca
135 | Vrba/srecna/draga/vas
148 | Vrba/srecna/vas
162 | srecna/vas
170 | Vrba/srecna/vas
177 | Vrba/srecna/draga/vas
183 | Vrba/srecna/draga/vas/domaca
195 | Vrba/srecna/draga/vas
208 | Vrba/srecna/vas
222 | srecna/vas
228 | srecna
240 |
```

Figure 1: Test case J9.

Submission

Submit your program as a single file named `DN09_`*`vvvvvvvv`*`.java`, where *vvvvvvvv* represents your student ID number.