The DOM(Document Object Model) is a way to represent HTML pages. The entire document is composed of nested elements. There is one parent element and all other elements are children or grandchildren of this element. In order to traverse between the elements, the browser uses few functions. In this problem we will be simulating a subset of those functions, on a given HTML document.

For this problem we will assume that the html document is defined as follows.
Each line of the document will be in one of the following format:

i) `<n value='`$<$ *word* $>$`'>` where $<$ *word* $>$ will consist of one or more alphabets.

ii) `</n>`

Here lines of first category indicate the opening of a new element and that of the second indicates the closing of one. Note that, an element may have zero or more elements nested inside so a `</n>` would indicate the closing of most recently opened element.

The simulation will be done as follows: Initially a pointer will be pointing the parent element. Each instruction will move the pointer to a new element, depending on the instruction given. There maybe situations when the instruction given will attempt to move the pointer to an element that doesn't exist. In that case, the pointer will remain at the current element.

Input

Each case of input starts with a positive number $N \leq 1000$. N lines then follow, each containing the definition of the document. You may assume that the definition is correct and complete. The next line will contain a positive integer $I \leq 100$, where I denotes the total number of instructions for that case. Next I lines each contain a single valid instruction. The last case is followed by a value of 0 for N. Each line of input file will contain at most 100 characters.

The instructions will be from the following set:

i) `first_child` : move to the first child of the current element.

ii) `next_sibling` : move to the next sibling of the current element.

iii) `previous_sibling` : move to the previous sibling of the current element.

iv) `parent` : move to the parent element of the current element.

Output

The output for each case starts with a line containing the case number in the form, '`Case` $< x >$`:`' , where $< x >$ denotes the case number. Each of the next I lines, one for each instruction, contain the value of the current node as described earlier. See the sample input/output for further clarification.

Sample Input

```
4
<n value='parent'>
<n value='child'>
</n>
</n>
2
next_sibling
first_child
0
```

Sample Output

```
Case 1:
parent
child
```