Problem E: Automatic Editing

Source file: `autoedit.{c, cpp, java, pas}`
Input file: `autoedit.in`
Output file: `autoedit.out`

Text-processing tools like *awk* and *sed* allow you to automatically perform a sequence of editing operations based on a script. For this problem we consider the specific case in which we want to perform a series of string replacements, within a single line of text, based on a fixed set of rules. Each rule specifies the string to find, and the string to replace it with, as shown below.

Rule	Find	Replace-by
1.	`ban`	`bab`
2.	`baba`	`be`
3.	`ana`	`any`
4.	`ba b`	`hind the g`

To perform the edits for a given line of text, start with the first rule. Replace the first occurrence of the *find* string within the text by the *replace-by* string, then try to perform the same replacement *again* on the new text. Continue until the *find* string no longer occurs within the text, and then move on to the next rule. Continue until all the rules have been considered. Note that (1) when searching for a *find* string, you always start searching at the beginning of the text, (2) once you have finished using a rule (because the *find* string no longer occurs) you never use that rule again, and (3) case is significant.

For example, suppose we start with the line

```
banana boat
```

and apply these rules. The sequence of transformations is shown below, where occurrences of a *find* string are underlined and replacements are boldfaced. Note that rule 1 was used twice, then rule 2 was used once, then rule 3 was used zero times, and then rule 4 was used once.

Before	After
`banana boat`	**bab**`ana boat`
`babana boat`	`ba`**bab**`a boat`
`bababa boat`	**be**`ba boat`
`beba boat`	`be`**hind the g**`oat`

The input contains one or more test cases, followed by a line containing only 0 (zero) that signals the end of the file. Each test case begins with a line containing the number of rules, which will be between 1 and 10. Each rule is specified by a pair of lines, where the first line is the *find* string and the second line is the *replace-by* string. Following all the rules is a line containing the text to edit. For each test case, output a line containing the final edited text.

Both *find* and *replace-by* strings will be at most 80 characters long. *Find* strings will contain at least one character, but *replace-by* strings may be empty (indicated in the input file by an empty line). During the edit process the text may grow as large as 255 characters, but the final output text will be less than 80 characters long.

The first test case in the sample input below corresponds to the example shown above.

Example input:

```
4
ban
bab
baba
be
ana
any
ba b
hind the g
banana boat
1
t
sh
toe or top
0
```

Example output:

```
behind the goat
shoe or shop
```