# Problem D
# 77377
# Input: press.txt

At the risk of its future, International Cellular Phones Corporation (ICPC) invests its resources in developing new mobile phones, which are planned to be equipped with Web browser, mailer, instant messenger, and many other advanced communication tools. Unless members of ICPC can complete this stiff job, it will eventually lose its market share.

You are now requested to help ICPC to develop intriguing text input software for small mobile terminals. As you may know, most phones today have twelve buttons, namely, ten number buttons from "0" to "9" and two special buttons "∗" and "#". Although the company is very ambitious, it has decided to follow today's standards and conventions. You should not change the standard button layout, and should also pay attention to the following standard button assignment.

| button | letters | button | letters |
|---|---|---|---|
| 2 | a, b, c | 6 | m, n, o |
| 3 | d, e, f | 7 | p, q, r, s |
| 4 | g, h, i | 8 | t, u, v |
| 5 | j, k, l | 9 | w, x, y, z |

This means that you can only use eight buttons for text input.

Most users of current ICPC phones are rushed enough to grudge wasting time on even a single button press. Your text input software should be economical of users' time so that a single button press is sufficient for each character input. In consequence, for instance, your program should accept a sequence of button presses "77377" and produce the word "press". Similarly, it should translate "77377843288866" into "press the button".

Ummm... It seems impossible to build such text input software since more than one English letter is represented by a digit! For instance, "77377" may represent not only "press" but also any one of 768 ($= 4 \times 4 \times 3 \times 4 \times 4$) character strings. However, we have the good news that the new model of ICPC mobile phones has enough memory to keep a dictionary. You may be able to write a program that filters out *false words*, i.e., strings not listed in the dictionary.

## Input

The input consists of multiple data sets, each of which represents a dictionary and a sequence of button presses in the following format.

$n$
$word_1$
$\vdots$
$word_n$
$sequence$

$n$ in the first line is a positive integer, representing the number of words in the dictionary. The next $n$ lines, each representing a word in the dictionary, only contain lower case letters from 'a' to 'z'. The order of words in the dictionary is arbitrary (not necessarily in the lexicographic order). No words occur more than once in the dictionary. The last line, *sequence*, is the sequence of button presses, and only contains digits from '2' to '9'.

You may assume that a dictionary has at most one hundred words and that the length of each word is between one and fifty, inclusive. You may also assume that the number of input digits in the *sequence* is between one and three hundred, inclusive.

A line containing a zero indicates the end of the input.

## Output

For each data set, your program should print all sequences that can be represented by the input sequence of button presses. Each sequence should be a sequence of words in the dictionary, and should appear in a single line. The order of lines does not matter.

Two adjacent words in a line should be separated by a single space character and the last word should be followed by a single period ('.').

Following those output lines, your program should also print a terminating line consisting solely of two hyphens ("--"). If there are no corresponding sequences of words, your program should only print the terminating line.

You may assume that for each data set the number of output lines is at most twenty, excluding the terminating line.

## Sample Input

```
5
push
press
the
button
bottom
77377843288866
4
i
am
```

```
going
go
42646464
3
a
b
c
333
0
```

## Output for the Sample Input

```
press the button.
--
i am going.
i am go go i.
--
--
```