UCF "Practice" Local Contest — August 27, 2011

T9 Craziness (filename: tnine)

Text-messaging is the current rage. Sitting in a boring class with nothing to do? Discreetly text your friend! The class isn't disturbed and you are no longer bored. It even allows you to make fun of that weird kid in class without actually saying a thing. But, as you may have noticed, regular texting, which includes pressing on each key up to four times to select the proper letter, gets tedious. Instead, you have discovered the T9 feature that allows you to press each corresponding numerical key once, but figures out exactly the word you meant to type, even though multiple letters correspond to each digit.

Here is exactly how T9 works: If you want to type a message to someone using a cell-phone pad, you only use the digits 2 through 9. The table below provides the corresponding letters for each digit:

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

If you wanted to type the message hello, you would just type 43556. Clearly there are many alphabetic strings that correspond to this five-digit string, but of all of those, hello is the only word. It turns out that for most words, no other word will correspond to the same exact set of digits. This is how T9 works – it cross references the digits pressed with a dictionary of words.

Obviously there are some instances where multiple words map to the same digits, such as "book" and "cool" which are both represented by "2665". In these cases, T9 selects the most frequently chosen word. (Phones with advanced T9 adapt to the user's typical word choices.)

For our purposes, your goal will be to write a program that translates messages typed in T9 to their alphabetic format. In the case that more than one message is possible, your program should simply calculate the total number of possible messages. (Note: This value should only be determined based upon the total number of possible combinations of words that could be formed and should not be restricted by the meaning of those words in any manner.) If no message is possible for the input T9 text, then your program should determine this as well.

The Problem:

Given a dictionary of valid words, and a set of messages sent in valid T9 format, determine the corresponding alphabetic message. If no message is possible, determine this. If multiple messages are possible, calculate how many are possible.

The Input:

The first line of the input file will contain a single integer n ($1 \leq n \leq 30000$), indicating the number of words in the dictionary. Each of the next n input lines will contain a single word from the dictionary. Each word will only consist of lowercase letters ($length \geq 1$) and there will be no duplicate words in the dictionary. Assume the input words start in column 1 and will not exceed column 80.

The next input line will contain a single integer, m, representing the number of messages to convert. Each of the next m lines will contain one message. A message will contain numeric strings (i.e., only digits 2 through 9) separated by spaces on a single line, with the first string starting in column 1 and each string being separated by a single space. No message will exceed 80 characters total, including spaces.

The Output:

For each input message, print a heading. Then, output one of the following three options:

If there is NO corresponding message, then print the following message out:

```
not a valid text
```

If there is exactly one possible message, then print out the corresponding message, all in lower case.

If there is more than one possible message, then print a message of the following format:

```
there are X possible messages
```

where X represents the total possible messages that could have been texted. You are guaranteed that X is less than $2^{31} - 1$.

Leave a blank line after the output for each data set. Follow the format illustrated in Sample Output.

(Sample Input/Output on the next page)

Sample Input:

```
11
hello
good
i
a
went
to
the
beach
cool
read
book
3
4 9368 86 843 23224
4 7323 2 2665 2665
8447 47 843 5278 8378 2273
```

Sample Output:

```
Message #1: i went to the beach

Message #2: there are 4 possible messages

Message #3: not a valid text
```