2335 T9

A while ago it was quite cumbersome to create a message for the Short Message Service (SMS) on a mobile phone. This was because you only have nine keys and the alphabet has more than nine letters, so most characters could only be entered by pressing one key several times. For example, if you wanted to type "`hello`" you had to press key 4 twice, key 3 twice, key 5 three times, again key 5 three times, and finally key 6 three times. This procedure is very tedious and keeps many people from using the Short Message Service.

This led manufacturers of mobile phones to try and find an easier way to enter text on a mobile phone. The solution they developed is called *T9 text input*. The "9" in the name means that you can enter almost arbitrary words with just nine keys and without pressing them more than once per character. The idea of the solution is that you simply start typing the keys without repetition, and the software uses a built-in dictionary to look for the "most probable" word matching the input. For example, to enter "`hello`" you simply press keys 4, 3, 5, 5, and 6 once.

Figure 8: The Number-keys of a mobile phone.

Of course, this could also be the input for the word "`gdjjm`", but since this is no sensible English word, it can safely be ignored. By ruling out all other "improbable" solutions and only taking proper English words into account, this method can speed up writing of short messages considerably. Of course, if the word is not in the dictionary (like a name) then it has to be typed in manually using key repetition again.

More precisely, with every character typed, the phone will show the most probable combination of characters it has found up to that point. Let us assume that the phone knows about the words "`idea`" and "`hello`", with "`idea`" occurring more often. Pressing the keys 4, 3, 5, 5, and 6, one after the other, the phone offers you "`i`", "`id`", then switches to "`hel`", "`hell`", and finally shows "`hello`".

Write an implementation of the *T9 text input* which offers the most probable character combination after every keystroke. The probability of a character combination is defined to be the sum of the probabilities of all words in the dictionary that begin with this character combination. For example, if the dictionary contains three words "`hell`", "`hello`", and "`hellfire`", the probability of the character combination "`hell`" is the sum of the probabilities of these words. If some combinations have the same probability, your program is to select the first one in alphabetic order. The user should also be able to type the beginning of words. For example, if the word "`hello`" is in the dictionary, the user can also enter the word "`he`" by pressing the keys 4 and 3 even if this word is not listed in the dictionary.

Input

The first line contains the number of scenarios.

Each scenario begins with a line containing the number w of distinct words in the dictionary ($0 \leq w \leq 1000$). These words are given in the next w lines in ascending alphabetic order. Every line starts with the word which is a sequence of lowercase letters from the alphabet without whitespace, followed by a space and an integer p, $1 \leq p \leq 100$, representing the probability of that word. No word will contain more than 100 letters.

Following the dictionary, there is a line containing a single integer m. Next follow m lines, each consisting of a sequence of at most 100 decimal digits 29, followed by a single 1 meaning "next word".

Output

The output for each scenario begins with a line containing '`Scenario #`i`:`', where i is the number of the scenario starting at 1.

For every number sequence s of the scenario, print one line for every keystroke stored in s, except for the 1 at the end. In this line, print the most probable word prefix defined by the probabilities in the dictionary and the T9 selection rules explained above. Whenever none of the words in the dictionary match the given number sequence, print '`MANUALLY`' instead of a prefix.

Terminate the output for every number sequence with a blank line, and print an additional blank line at the end of every scenario.

Sample Input

```
2
5
hell 3
hello 4
idea 8
next 8
super 3
2
435561
43321
7
another 5
contest 6
follow 3
give 13
integer 6
new 14
program 4
5
77647261
6391
4681
26684371
77771
```

Sample Output

```
Scenario #1:
i
id
hel
hell
hello

i
```

```
id
ide
idea


Scenario #2:
p
pr
pro
prog
progr
progra
program

n
ne
new

g
in
int

c
co
con
cont
anoth
anothe
another

p
pr
MANUALLY
MANUALLY
```