Dependent Changes

The software project on which you are working is under a source code management system where every change is tracked with a Change ID (CID). Commands provided by the system allow versions of the source containing specific sets of changes to be retrieved. In particular, the `getversion` command lets you specify a list of CIDs, and it will determine all the source files affected by these CIDs and fetch those files with the appropriate changes applied. Unfortunately, this source management system is old and the `getversion` command tends to break if too many CIDs are given to it at once, yet monthly load builds often require large sets of CIDs. Until you can get this command updated, you need to work around its limitations by using several executions of `getversion` on smaller sets of CIDs. Ideally, you would like to execute `getversion` for each CID individually. But two or more CIDs may affect the same file, so in this case the CIDs must be retrieved together to get the effects of all changes - the same CID could not be given to two separate executions of `getversion` and still produce a consistent version. Your job is to write a program that analyzes the CID-file relationship and determine the largest number of `getversion` commands that will yield a consistent version.

The first line of input to this problem will be an integer n indicating a number of sets of CIS-file tuples to follow.

Each tuple for a set will be on a line by itself and consists of a CID and a file that it affects with these values separated by a colon. Blank lines will separate the sets. There will be no more than 1000 tuples per set. The output should consist of one line for each set, indicating the largest number of `getversion` commands that the set of CIDs could be spread over.

Sample Input

```
2
cid1:fileA
cid1:fileB
cid2:fileC
cid3:fileB
cid3:fileD
cid4:fileC
cid5:fileD
cid4:fileE
cid5:fileF

cid13:fileA
cid11:fileB
cid9:fileC
```

Sample Output

```
The group 1 CIDs can be broken into no more than 2 commands
The group 2 CIDs can be broken into no more than 3 commands
```

In the first set of this example, `cid1`, `cid3`, and `cid5` must be grouped together because they affect files `A`,`B`,`D` and `F`; and `cid2` and `cid4` must be grouped together because they affect files `C` and `E`, giving a maximum of two `getversion` commands. In the second set, there are no common files, so a separate `getversion` command could be used with each CID.

JumpShip

You pilot a 25th Century *jumpship*. These vehicles have the ability to leap between *jumppairs*, namely ordered pairs of *jumpports* which may be separated by astronomical distances but are adjacent in unidirectional hyperspace. Thus, jumps take a variable amount of time, which can even be negative.

Your goal is to a plot a *jumppath* that minimizes travel time. In fact, there is particular advantage to making the voyage in negative time since the effect on our bodies is to reverse the aging process by the corresponding number of days; with such a course, you can collect a proportional surcharge on passenger tickets.

However, as forecast in 20th-21st century science fiction, the paradox resulting from a return to a location before one leaves that location could be disastrous. A *jumpport* that admits such a path is marked "parodox-prone". Naturally, the tight controls maintained by the Federal Jumping Agency prohibit jumping out of any port if there is any jumppath from there to a paradox-prone port, even if the intention was to go toward a different and entirely safe place.

In this problem, you are asked to write a program for determining optimal jumppaths, given the usable jumppairs and their corresponding time deviations.

The first line of the input will be an integer n indicating the number of instances of jumppaths to construct. Each instance will begin with a line indicating the number m of jumppairs, identified by their 3-letter jumpport codes, in the sector followed by the departure port and then the destination port (separated by single spaces). Each of the next m lines will be of the form `ABC XYZ` t signaling the possibility of a jump from `ABC` to `XYZ` in time t (t will be a integer of absolute value ≤ 1000).

For each instance where jump travel is safe, you should indicate an optimal jumppath, along with the total time, using the format

$jpcode_1\ time_1\ jpcode_2\ time_2\ \ldots\ time_{\ell-1}\ jpcode_\ell$ `for total time` T

where time$_i$ is the time deviation in jumping from $jpcode_i$ to $jpcode_{i+1}$.

However, if a paradox-prone port is reachable from the departure point, you should indicate

```
Potential paradox, jumping prohibited.
```

The instances should be separated by a blank line in the output.

(See sample Input/Output on reverse.)

Sample Input

```
3
11 ACT HPR
ACT BIT 3
ACT DSL 5
ACT FAQ 2
BIT CAD -4
CAD HPR 4
DSL EOT 6
EOT DSL -3
EOT HPR 8
FAQ GIF 3
GIF FAQ -2
GIF HPR 7
3 EUG SFO
EUG PDX -3
EUG SFO 2
PDX SFO 1
7 COM EDU
COM GOV -1
GOV EDU -1
COM MIL 5
MIL NET 2
NET ORG -1
ORG INT -1
INT MIL -1
```

Sample Output

```
ACT 3 BIT -4 CAD 4 HPR for total time 3

EUG -3 PDX 1 SFO for total time -2

Potential paradox, jumping prohibited.
```

Anagram Anyone?

String S is an anagram of string T if it contains the same letters with the same multiplicities. In other words, an anagram of a string is a permutation of the letters in the string. For example, if S contains two occurences of the letter `x`, then T must also contain exactly two occurences of the letter `x`. It is customary to ignore spaces when making an anagram of a phrase: “dirty room” and “dormitory” are anagrams of each other, even though “dirty room” contains a space and “dormitory” does not. It is also customary to insist that the solution to an anagram puzzle be words from a dictionary.

In this problem, you are given a number of input phrases consisting of one or more words (sequences of non blank characters, not necessarily in any dictionary). Your program must produce an output phrase that is an anagram of the input if one can be constructed; this output phrase must consist of exactly one or two words chosen from a provided list of common English words.

A list of 999 common English words, one per line, is available as `words.txt`.

The first line of input to the program is the number of phrases to follow. Phrases consist of one to three strings totaling no more than 20 characters, separated by blanks, and made up of lower case letters.

For each input phrase, your program must produce an output phrase of one or two words from the word list, where the letters of these words are the same letters as the input phrase letters. If there is no possible anagram phrase, your program should print

```
There is no anagram.
```

Sample Input

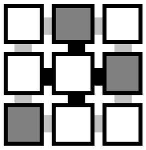
```
4
pinot noir
contest
ska
pizza pie
```

Sample Output

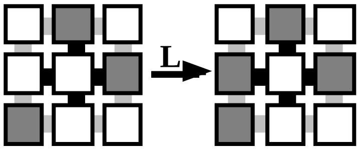
```
iron point
cost ten
ask
There is no anagram.
```

ReversaTile

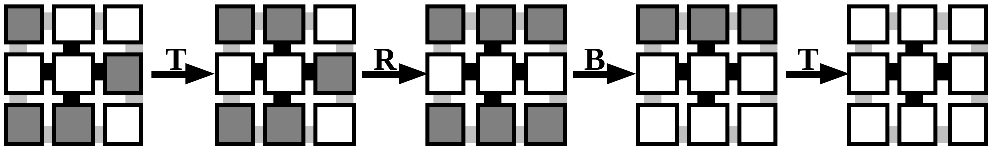
ReversaTile is a mechanical puzzle consisting of 9 square tiles, each white on one face and black on the reverse. The tiles are linked together in a 3 × 3 formation:



The ingenious linkage allows for the outside rows (left, right, top, bottom) to be pivoted around the middle tile of the row. For example, the operation **L** switches the upper left tile with the lower left tile with all three of the tiles flipped over:



Similarly operations **R, T, B** privot the right, top, bottom rows, respectively. The puzzle is solved when all tiles show their white face. For example:



In this problem, you are asked to write a program that solves the puzzle from a given configuration using no more than 10 moves (if that is possible).

The first line of input will consist of an integer n, indicating the number of puzzle instances to follow. For each, the starting configuration will be described in 3 lines indicating the color of the tiles (b or w) on that line (with no spaces in between).

Your output for each puzzle instance should be either the phrase

```
There is no solution within 10 moves.
```

or a sequence of configurations from the initial one to the solved state; this should occupy just three lines with the configurations lined up horizontally and separated by triple spaces for readability (as shown in the samples on the reverse side of this page): the first line should show the sequence for the top row of the puzzle, etc. Skip a line between solutions.

(See sample Input/Output on reverse side.)

Sample Input

```
4
bww
wwb
bbw
bbb
bbb
bbb
wbb
www
bww
wwb
wwb
wwb
```

Sample Output

```
bww   bbw   bbb   bbb   www
wwb   wwb   www   www   www
bbw   bbw   bbb   www   www

There is no solution within 10 moves.

There is no solution within 10 moves.

wwb   www
wwb   www
wwb   www
```

Possible Products

If a binary operation • is not associative then there is some ambiguity about the value of unparenthesized "products". For example, consider the operation • on the set $\{A, B, C\}$ defined by the "multiplication table":

$\bullet$	A	B	C
A	C	C	A
B	B	C	B
C	B	B	A

Thus, $A{\bullet}A = C$, $A{\bullet}B = C$, $A{\bullet}C = A$, $B{\bullet}A = B$, etc. Then the string $C{\bullet}A{\bullet}B{\bullet}C$ has five possible "products".

$$\begin{aligned}
C{\bullet}(A{\bullet}(B{\bullet}C)) &= C{\bullet}(A{\bullet}B) = C{\bullet}C = A \\
C{\bullet}((A{\bullet}B){\bullet}C) &= C{\bullet}(C{\bullet}C) = C{\bullet}A = B \\
(C{\bullet}A){\bullet}(B{\bullet}C) &= B{\bullet}(B{\bullet}C) = B{\bullet}B = C \\
((C{\bullet}A){\bullet}B){\bullet}C &= (B{\bullet}B){\bullet}C = C{\bullet}C = A \\
(C{\bullet}(A{\bullet}B)){\bullet}C &= (C{\bullet}C){\bullet}C = A{\bullet}C = A
\end{aligned}$$

Three of the interpretations yield the result A, one yields B and one yields C.

In this problem, you are asked to write a program to determine how many times each element of a set arises in the interpretation of a given string according to a given operation.

The first line of input to this problem will consist of an integer n, indicating the number of problem instances to follow. Each instance with start with a line indicating the number m, with $m \leq 7$ of elements in the set, which will always consist of the initial m (capital) letters of the alphabet $A, B, C, \ldots$ in order. The next m lines, with m letters per line (separated by single spaces) will comprise the table for the operation, e.g., if the 3rd character of the second line is E then $B \circ C = E$. Finally, one more line will give the string $\$$ under investigation, given as a sequence of set elements of length at most 25, separated by single spaces.

Your output for each problem should be on a single line that lists the m elements of the set, each followed by the number of ways the product along $\$$ could have been interpreted as that element. There should be a space between entries on that line.

Sample Input

```
2
3
C C A
B C B
B B A
C A B C
2
A A
A A
B A B
```

Sample Output

```
A 3 B 1 C 1
A 2 B 0
```