Phone Words

Businesses often like to encode their phone numbers by an easily remembered word using the matching of letters to digits on the phone dial:

	A B C	D E F	G H I	J K L	M N 0	P R S	T U V	W X Y	
1	2	3	4	5	6	7	8	9	0

For this problem, you are required to find the words that encode a given 7-digit phone number. The words are to be found in a dictionary which is given as an alphabetically sorted ASCII file of 7-letter words (one word per line) in `local/contest/data/phonedict`. If there is no word that corresponds to the given phone number, then your program should discover that.

The input will start with a single integer m on a line indicating the number of phone numbers to follow. The next m lines will give the phone numbers, each in the standard form \$\$\$-\$\$\$\$. The corresponding m lines of output should list words for each phone number, separated by a single space if there is more than one word (there will not be more than 5 words for any number), or else the phrase

`There is no word for the number` \$\$\$-\$\$\$\$

(indicating the input number).

Sample Input

```
4
266-7453
678-1234
736-6268
242-6868
```

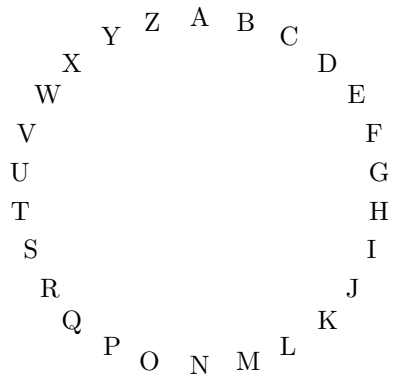
Sample Output

```
compile
There is no word for 678-1234
pennant remnant
There is no word for 242-6868
```

Roman RSA

The ancient Romans used the following system for encrypting messages:

They arranged the letters of the alphabet clockwise in a circle and they chose a positive integer k. Then, for each letter in the message, they started in their alphabet-circle from that letter, counted clockwise k letters, and they wrote down the letter where they finished counting. For example, if $k = 3$, A is encoded by D, B is encoded by E, etc. The encrypted message was then sent to a recipient who, knowing k, could easily decipher the message by counting counterclockwise k letters from encrypted letters.



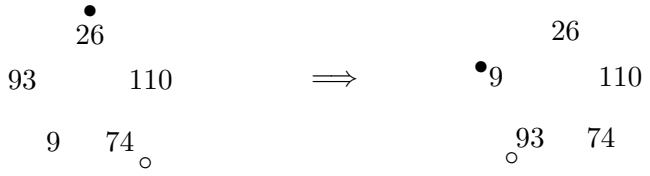
Unfortunately, the scheme was not very secure since their enemies just had to try a few values of k to decrypt the message.

Called upon to devise safer encryption, the Roman Security Association (RSA) enlisted its chief theorist Rivestus. He suggested the following: The arrangement of letters seems to be a good idea, anyway we have already manufactured all those alphabet circles. However, we can have fast encryption without keeping to a constant k. Here is how we can do this –

At the start, we shall have 5 positive integers written on stones arranged in a circle, along with a white pebble next to one stone and a black pebble next to another. (The recipient of the message will know this initial configuration.) For each letter of the message

1. Move the white pebble by one location clockwise.
2. Move the black pebble clockwise by as many locations as the number on the stone next to the white pebble.
3. Swap the positions of the stones marked by the pebbles, but not the pebbles (do nothing if the pebbles are next to the same stone).

   Here is an example of the effect of these three steps on the stone/pebble configuration:



4. Set k equal to the sum of the values on the stones that are now next to the pebbles, in particular, if the pebbles are located at the same stone, take twice the value of that stone. (In the above example, $k = 102$.) Use this k to encrypt the current letter.

If one knows the *starting* configuration of the stones and pebbles, the message can be decrypted by the same process, except that one counts counterclockwise on the letter circle.

This worked well until one day, just after the legion commander encrypted a report to the emperor, the Gauls led by Shamirix raided the camp, and obtained the encrypted message. In the excitement, it did not occur to the commander to deal with the stones and pebbles which were still in the configuration reached at the end of the encryption process. A talented theorist himself, Shamirix realized he could use this *final* configuration to decode the message.

For this problem, you are to write a program which, given the stone/pebble configuration after encrypting, can be used to decrypt the encrypted message.

Each instance of the problem is described on three lines:

- 1st *line:* 5 integers s_1 s_2 s_3 s_4 s_5, separated by single spaces, indicating in order the numbers on the stones.
- 2nd *line:* 2 integers W B between 1 and 5, separated by a single space, indicating the position of the stones next to the white and black pebbles.
- 3rd *line:* the encrypted message (at most 50 characters in length with no spaces).

The first line of the input to your program will be a number m indicating the number of decryption instances to follow. The next $3m$ lines will describe the m instances as indicated above.

Output for each instance should be the decrypted message.

Sample Input

```
2
5 6 2 7 10
2 3
RSA
1 3 7 5 2
1 1
LUIAOOU
```

Sample Output

```
CIS
GODUCKS
```

Word Chains

This problem is based on challenges often posed on *NPR Weekend Edition*. Given two words A and B, find the shortest sequence of words $W_1, W_2, \cdots, W_{n-1}, W_n$ such that:

- $W_1 = A$ and $W_n = B$
- each W_i is an uncapitalized word in a standard dictionary. (For our purposes, the "dictionary" will be a text file containing a list of words.)
- each consecutive pair W_i, W_{i+1} differ by substitution of one character in a single position.

The dictionary for this exercise will be found in `local/contest/data/chaindict`. It is a text file containing words, one word per line, each word consisting of exactly four lower-case letters.

Your program should take two words of exactly four lower case letters and form a word chain as above. For the test instances we provide, we guarantee that it is possible to form a word chain no longer than seven words long using words from our dictionary. (There may be more than one acceptable solution.)

The input to the problem will start with a single integer m on a line indicating the number of test instances to follow. The following m lines will each give a pair of four-letter words, a starting word for the chain and a target word, separated by a single space.

The output for each instance should consist of a chain, words from the start word to the target word, on a single line with one space between words.

Sample Input

```
3
flip flop
real char
hack code
```

Sample Output

```
flip flop
real meal meat moat coat chat char
hack hark hare care core code
```

(See the reverse of this page for some help in reading the dictionary.)

Reading the dictionary.

The dictionary file contains less than 2000 words of exactly four letters, all lower case, in alphabetical order. Thus you will not need to implement particularly efficient data structures to hold the dictionary structure in main memory.

Those using Java may find the following example code useful. The constructor loads a dictionary file into a hash table. The `isPresent(s: String)` method tests for membership.

```
import java.io.*;
import java.util.Hashtable;

public class Dict {
     private String path = "./smalldict"; // Change to correct file
     private Hashtable dict;

     public Dict()  throws java.io.IOException {
        StreamTokenizer in;
        dict = new Hashtable(5000);
        try {
            FileInputStream f = new FileInputStream(path);
            Reader r = new BufferedReader(new InputStreamReader(f)); j
            in = new StreamTokenizer(r);
        } catch (IOException e) {
            System.err.println("Couldn't open " + path );
            return;
        }
        while (in.nextToken() != StreamTokenizer.TT_EOF)    {
                if (in.ttype == StreamTokenizer.TT_WORD) {
                    dict.put(in.sval, in.sval);
                } else {
                    System.err.print("Unrecognized token: " + in.ttype + " ");
                    System.err.println("Token value: " + in.sval);
                }
            } // while
    }// constructor Dict()

     /** Test for presence */
     public boolean isPresent(String s) {
         return dict.containsKey(s);
     }
}
```

Denali Deliveries

Trade in the Denali sector is expensive. The Hentai Dominion enjoys a monopoly on sector trade and offers only limited direct connections between planets at exhorbitant one-way fees.

Your company, Intragalactic Bark & Mulch, would like to set up operations delivering their unique dilithium fertilizer throughout the system. Because of the restricted availability of interstellar craft, the company plans to build a warehouse on just one planet in each star system to ship and store all their product. Cargo will be sent monthly from this central warehouse to each of the retail outlets on all other planets in that system. Because of the instability of compressed dilithium, the monthly shipment to a retail outlet is sealed in a single antimatter container that cannot be unsealed until it reaches its destination planet; thus shipments to various retail outlets cannot be combined enroute even if they are using some of the same connections.

Your task is to choose the planet in each star system such that your total of the monthly delivery costs from the central warehouse to the retail outlets is minimized and to determine the planetary connections that would be used for these deliveries.

The Dominion schedule of fees for each star system is coded as follows:

> A line with a single integer m indicating the number of planets in the system. Then m lines, one for each planet in the system, each in the form:
>
> $$\mathcal{P}\ q\ \mathcal{D}_1\ f_1\ \mathcal{D}_2\ f_2\ \cdots\ \mathcal{D}_q\ f_q$$
>
> where $\mathcal{P}$ is the name of a planet, q (with $1 \le q \le m$) is the number of direct Dominion trade connections out of $\mathcal{P}$ and then, for each of these connections, the name $\mathcal{D}_i$ of the destination planet and the fee f_i in denalians for transport of a single container along this leg. There is a single space between items on the line. Note that the fee for direct transport from planet $\mathcal{A}$ to planet $\mathcal{B}$ may not be the same as the fee from $\mathcal{B}$ to $\mathcal{A}$; indeed, the Hentai may not even offer a direct return connection.

The input to your program will be the Hentai table of delivery fees in the Denali sector. The first line will be an integer n indicating the number of star systems to follow, then n system schedules as above.

The first line of the output for each system should indicate the name of the warehouse planet: the warehouse needs to be chosen so as to minimize the *sum* of the Hentai fees to deliver cargo from the warehouse to the other planets in the system; clearly you need to choose *each* route to a retail outlet so as to minimize the fee for *that* delivery. The succeeding lines should indicate just the Hentai connections that will be utilized in delivery routes from the warehouse to the retail outlets; from each planet $\mathcal{A}$ indicate the utilized connections from $\mathcal{A}$ in the form

$$\mathcal{A}\ \texttt{to}\ \mathcal{B}\ \texttt{[and}\ \mathcal{C}\texttt{]}\ \texttt{[and}\ \mathcal{D}\texttt{]}\ \cdots$$

(For our test data, the output for each planet will fit on a single line.) If no connections are used out of a planet, it need not be so listed.

There should be one blank line between systems.

(Sample input/output on reverse)

Sample Input

```
3
4
Alderaan 2 Endor 100 Hoth 40
Hoth 2 Endor 10 Naboo 10
Endor 2 Alderaan 30 Naboo 40
Naboo 1 Alderaan 50
3
Tinker 2 Evers 30 Chance 60
Evers 2 Chance 20 Tinker 70
Chance 2 Tinker 40 Evers 50
4
Groucho 1 Chico 10
Chico 1  Harpo 10
Harpo 1 Zeppo 20
Zeppo 3 Groucho 20 Chico 20 Harpo 20
```

Sample Output

```
Hoth
Hoth to Endor and Naboo
Endor to Alderaan

Tinker
Tinker to Evers
Evers to Chance

Zeppo
Zeppo to Groucho and Chico and Harpo
```

Modular Quadratics

This problem involves that old high school algebra exercise, solving quadratic equations.

$$ax^2 + bx + c = 0$$

But you do not have to worry about those messy square roots and such because you will only need to find the *integer* solutions. Moreover, to make life really easy, you will only need to make $ax^2 + bx + c$ *congruent** *to* 0 modulo a given power of 10.

The first line of input to this problem will consist of an integer m, indicating the number of instances to follow. Each of the following m lines will specify a sequence of four integers

$$a\ b\ c\ k$$

separated by single spaces, with $1 \leq k \leq 9$ and $0 \leq a, b, c < 10^k$.

Your output should be the list of all integers x, with $0 \leq x < 10^k$, such that

$$ax^2 + bx + c \equiv 0 \mod (10^k).$$

The solutions, if more than one, should be written one to a line. If there are no solutions, then your output should read:

```
There are no solutions.
```

Skip a line between the solutions to different instances.

Sample Input

```
4
1 2 0 1
1 0 4 2
2 2 1 3
3 2 2 1
```

Sample Output

```
0
8

14
36
64
86

There are no solutions.

8
```

*We say that

a is congruent to b modulo m or $a \equiv b \mod (m)$

if $a - b = qm$ for some integer q.