NOIP 普及组复赛
冲刺赛（11 天）

Problem 1

Larry graduated this year and finally has a job. He's making a lot of money, but somehow never seems to have enough. Larry has decided that he needs to grab hold of his financial portfolio and solve his financing problems. The first step is to figure out what's been going on with his money. Larry has his bank account statements and wants to see how much money he has. Help Larry by writing a program to take his closing balance from each of the past twelve months and calculate his average account balance.

Input Format:

The input will be twelve lines. Each line will contain the closing balance of his bank account for a particular month. Each number will be positive and displayed to the penny. No dollar sign will be included.

Output Format:

The output will be a single number, the average (mean) of the closing balances for the twelve

months. It will be rounded to the nearest penny, preceded immediately by a dollar sign, and followed by the end-of-line. There will be no other spaces or characters in the output.

Sample Input:

100.00

489.12

12454.12

1234.10

823.05

109.20

5.27

1542.25

839.18

83.99

1295.01

1.75

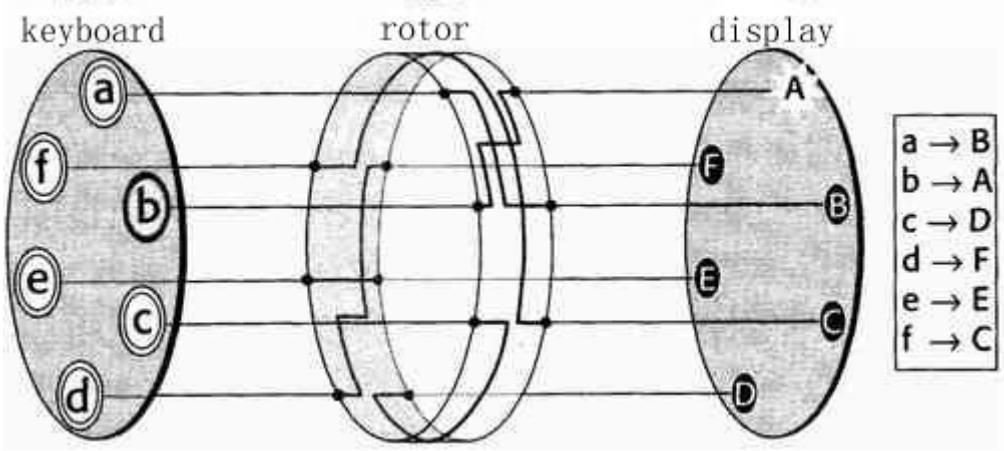
Sample Output:

$1581.42

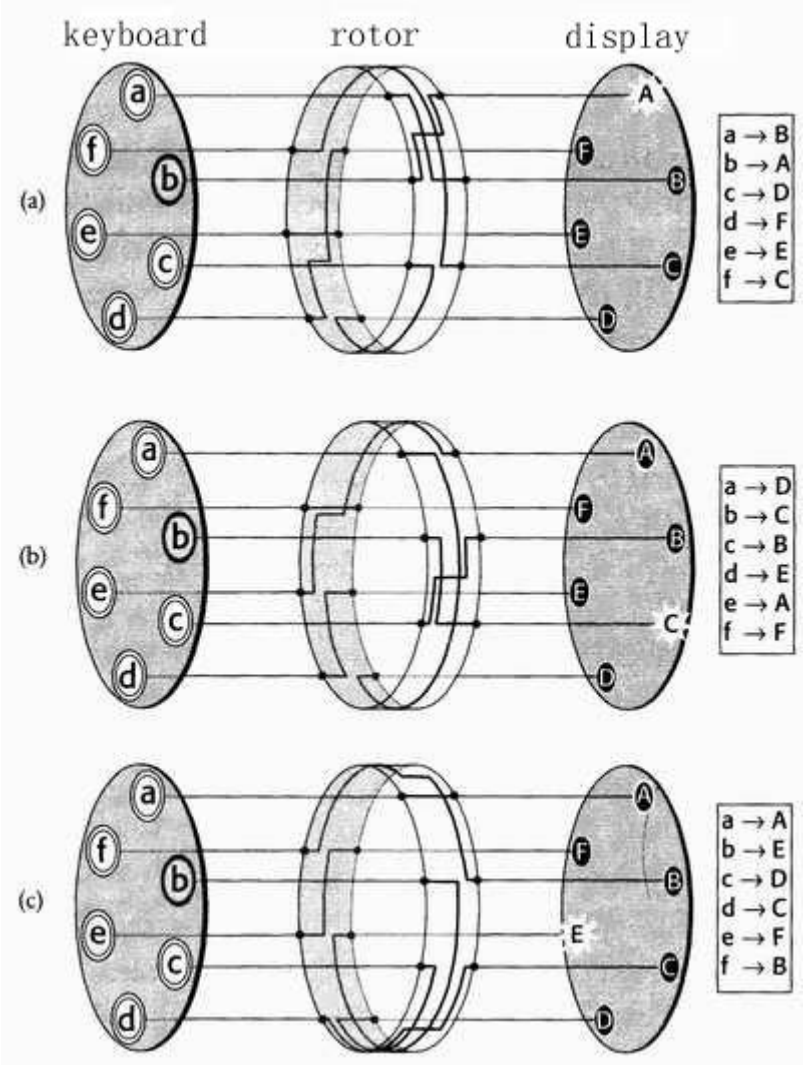
Problem 2

In World War II, Germany once used an electronic encryption machine called Enigma, which played a decisive role in the initial victories of Nazi Germany. It was proved to be one of the most reliable encryption systems in history. However, it was the blind trust on the reliability of the machine that brought about the doom of its user.聽

The structure of a one-rotor Enigma is shown as follows (the Enigma has only six keys):

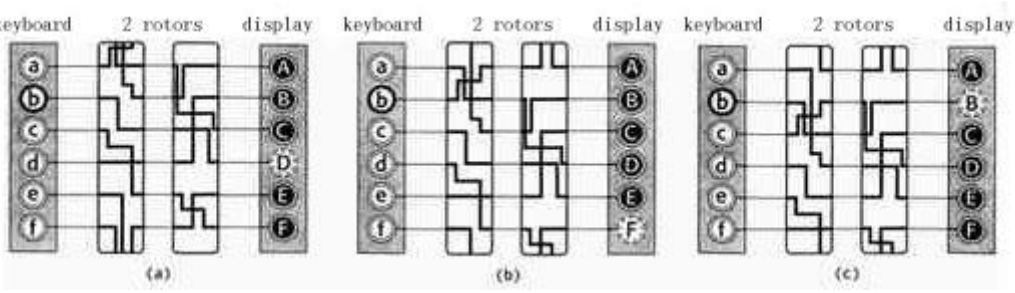


The key element of the Enigma is the rotor, as shown in the second figure, which uses electronic circuits to transform plaintext (input from keyboard) into cryptograph (output on screen). When one key on the keyboard is pressed, the corresponding cryptograph is shown on screen. Then the rotor will automatically revolve a one-letter-step to a different position. The following figures illustrate how the rotor works when letter "b" is pressed three successively times:



When letter "b" is pressed for the first time, the signal goes through the circuit and "A" is shown on screen. When the key is released, the rotor revolves one-letter-step to a different position that changes all the corresponding circuits so that each letter now has a different cryptograph. When letter "b" is pressed for the second time, the corresponding cryptograph is "C". So when letter "b" is pressed for the third time, the cryptograph is "E" according to the principle specified above.

Now the following figure shows the structure of a two-rotor Enigma.



The difference is that when a key is released, the second rotor won't revolve a step until the first one has finished one circle and returns to the original position. This is also the same in the case of three-rotor Enigma. That is: Only after the first rotor has finished one circle and return to the initial status, the second rotor will revolve a step. And only after the second rotor has finish one circle, the third rotor will revolve a step.

However, how did the Allied Forces obtain the information encrypted by Enigma? A person named Hans-Thilo Schimdt was very essential. He acted as a spy and provided the initial status of the three rotors in each Enigma to the Allied Forces once a month. The Allied Forces thus got everything they wanted by deciphering the intercepted cryptograph using the information offered by the spy.

Now, please design a program to obtain the plaintexts using the information offered by the Allied Forces.

Input

The input file contains several test cases representing several three-rotor Enigmas. The last test case in the input file is followed by a line containing a number 0.

Each case begins with a line containing an integer m (1 <= m <= 26) which indicates the number of sequential letters each rotor has. The first letter will always be A. (for example, m = 6 tells each rotor has 6 keys from A to F). The following three lines describe the initial status of the three rotors respectively. Each of them contains a string consisting of m capital character. For instance, a rotor with the initial status "BADFEC" indicates that the initial encrypt mechanism is to convert "abcdef" to "BADFEC", that is, original letter "a" corresponding to cryptograph letter "B", "b" to "A", "c" to "D", "d" to "F", "e" to "E" and "f" to "C". The forth line of each case contains an integer n which tells the number of cryptographs generated by the above Enigma. Then the following n lines are the n cryptographs respectively, which consist of m capital characters each.

Output

For each test case, the output should consist of two parts. The first line is the number of Enigma and a colon. The following lines are the plaintexts deciphered from the corresponding cryptographs. Each plaintext should be printed in one line. Note: The characters in the plaintext should be converted to the corresponding lowercases before they are printed.

Insert a blank line between test cases.

Sample Input

6

BADFEC

ABCDEF

ABCDEF

1

ACE

0

Output for the Sample Input

Enigma 1:

bbb

Problem 3

How can anagrams result from sequences of stack operations? There are two sequences of stack operators which can convert TROT to TORT:

[

i i i i o o o o

i o i i o o i o

]

where *i* stands for Push and *o* stands for Pop. Your program should, given pairs of words produce sequences of stack operations which convert the first word to the second.

Input

The input will consist of several lines of input. The first line of each pair of input lines is to be considered as a source word (which does not include the end-of-line character). The second line (again, not including the end-of-line character) of each pair is a target word. The end of input is marked by end of file.

Output

For each input pair, your program should produce a sorted list of valid sequences of *i* and *o* which produce the target word from the source word. Each list should be delimited by

[

]

and the sequences should be printed in "dictionary order". Within each sequence, each *i* and *o* is followed by a single space and each sequence is terminated by a new line.

Process

A stack is a data storage and retrieval structure permitting two operations:

Push - to insert an item and
Pop - to retrieve the most recently pushed item

We will use the symbol *i* (in) for push and *o* (out) for pop operations for an initially empty stack of characters. Given an input word, some sequences of push and pop operations are valid in that every character of the word is both pushed and popped, and furthermore, no attempt is ever made to pop the empty stack. For example, if the word FOO is input, then the sequence:

i i o i o o is valid, but
i i o is not (it's too short), neither is
i i o o o i (there's an illegal pop of an empty stack)

Valid sequences yield rearrangements of the letters in an input word. For example, the input word FOO and the sequence i i o i o o produce the anagram OOF. So also would the sequence i i i o o o. You are to write a program to input pairs of words and output all the valid sequences of i and o which will produce the second member of each pair from the first.

Sample Input

madam

adamm

bahama

bahama

long

short

eric

rice

Sample Output

[

i i i i o o o i o o

i i i i o o o o i o

i i o i o i o i o o

i i o i o i o o i o

]

[

i o i i i o o i i o o o

i o i i i o o o i o i o

i o i o i o i i i o o o

i o i o i o i o i o i o

]

[

]

[

i i o i o i o o

]

Problem 4

On every June 1st, the Children's Day, there will be a game named "crashing balloon" on TV. The rule is very simple. On the ground there are 100 labeled balloons, with the numbers 1 to 100. After the referee shouts "Let's go!" the two players, who each starts with a score of "1", race to crash the balloons by their feet and, at the same time, multiply their scores by the numbers written on the balloons they crash. After a minute, the little audiences are allowed to take the remaining balloons away, and each contestant reports his\her score, the product of the numbers on the balloons he\she's crashed. The unofficial winner is the player who announced the highest score.

Inevitably, though, disputes arise, and so the official winner is not determined until the disputes are resolved. The player who claims the lower score is entitled to challenge his\her opponent's score. The player with the lower score is presumed to have told the truth, because if he\she were to lie about his\her score, he\she would surely come up with a bigger better lie. The challenge is upheld if the player with the higher score has a score that cannot be achieved with balloons not crashed by the challenging player. So, if the challenge is successful, the player claiming the lower score wins.

So, for example, if one player claims 343 points and the other claims 49, then clearly the first player is lying; the only way to score 343 is by crashing balloons labeled 7 and 49, and the only way to score 49 is by crashing a balloon labeled 49. Since each of two scores requires crashing the balloon labeled 49, the one claiming 343 points is presumed to be lying.

On the other hand, if one player claims 162 points and the other claims 81, it is possible for both to be telling the truth (e.g. one crashes balloons 2, 3 and 27, while the other crashes balloon 81), so the challenge would not be upheld.

By the way, if the challenger made a mistake on calculating his/her score, then the challenge would not be upheld. For example, if one player claims 10001 points and the other claims 10003, then clearly none of them are telling the truth. In this case, the challenge would not be upheld.

Unfortunately, anyone who is willing to referee a game of crashing balloon is likely to get over-excited in the hot atmosphere that he\she could not reasonably be expected to perform the intricate calculations that refereeing requires. Hence the need for you, sober programmer, to provide a software solution.

Input

Pairs of unequal, positive numbers, with each pair on a single line, that are claimed scores from a game of crashing balloon.

Output

Numbers, one to a line, that are the winning scores, assuming that the player with the lower score always challenges the outcome.

Sample Input

343 49

3599 610

62 36

Sample Output

49

610

62