

# Problem 1. References

(Time Limit: 2 seconds)

## Problem Description

Editors of an electronic magazine make draft versions of the documents in the form of text files. However, publications should meet some requirements, in particular, concerning the rules of reference use. Unfortunately, lots of the draft articles violate some rules. It is desirable to develop a computer program that will make a publication satisfy all the rules from a draft version.

Let's call a "paragraph" a set of lines in the article going one after another, so that paragraphs are separated by at least one empty line (an "empty line" is a line that containing no characters different from spaces). Any paragraph can contain an arbitrary number of references. A reference is a positive integer not greater than 999 enclosed in square brackets (for example: [23]). There will be no spaces between the brackets and the number. The square brackets are not used in any other context but reference.

There can be two types of paragraph - "regular" and "reference description". Reference description differs from the regular paragraph because it begins with the reference it describes, for example:

[23] It is the description ...

The opening square bracket will be at the first position of the first line of the "reference description" paragraph (i.e. there will be no spaces before it). No reference description paragraph will contain references inside itself.

Each reference will have exactly one corresponding description and each description will have at least one reference to it.

To convert a draft version to a publication you have to use the following rules.

- References should be renumbered by the successive integer numbers starting from one in the order of their first appearance in the regular paragraphs of the source draft version of the document.
- Reference descriptions should be placed at the end of the article ordered by their number.
- The order of "regular" paragraphs in the document should be preserved.

- Your program should not make any other changes to the paragraphs.

## Input Format

The input will be a text containing a draft article your program should process. All lines will be no more than 80 characters long. Any reference description will contain no more than 3 lines. The input file will contain up to 40000 lines.

## Output Format

The output contains the result of processing. All paragraphs should be separated by one "true" empty line (i.e. a line that contains no characters at all). There should be no empty lines

## Example

Sample Input:	Sample Output:
[5] Brownell, D, "Dynamic Reverse Address Resolution Protocol(DRARP)", Work in Progress.	The Reverse Address Resolution Protocol (RARP) [1] (through the extensions defined in the Dynamic RARP (DRARP) [2]) explicitly addresses the problem of network address discovery, and includes an automatic IP address assignment mechanism.
The Reverse Address Resolution Protocol (RARP) [10] (through the extensions defined in the Dynamic RARP (DRARP) [5]) explicitly addresses the problem of network address discovery, and includes an automatic IP address assignment mechanism.	The Trivial File Transfer Protocol (TFTP) [3] provides for transport of a boot image from a boot server. The Internet Control Message Protocol (ICMP) [4] provides for informing hosts of additional routers via "ICMP redirect" messages.
[10] Finlayson, R., Mann, T., Mogul, J., and M. Theimer, "A Reverse Address Resolution Protocol", RFC 903, Stanford, June 1984.	Works [1], [4] and [3] can be obtained via Internet.
[16] Postel, J., "Internet Control Message Protocol", STD 5, RFC 792, USC/Information Sciences Institute, September 1981.	[1] Finlayson, R., Mann, T., Mogul, J., and M. Theimer, "A Reverse Address Resolution Protocol", RFC 903, Stanford, June 1984.
The Trivial File Transfer Protocol (TFTP) [20] provides for transport of a boot image from a boot server. The Internet Control Message Protocol (ICMP) [16] provides for informing hosts of additional routers via "ICMP redirect" messages.	[2] Brownell, D, "Dynamic Reverse Address Resolution Protocol(DRARP)", Work in Progress.
[20] Sollins, K., "The TFTP Protocol (Revision 2)", RFC 783, NIC, June 1981.	[3] Sollins, K., "The TFTP Protocol (Revision 2)", RFC 783, NIC, June 1981.
Works [10], [16] and [20] can be obtained via Internet.	[4] Postel, J., "Internet Control Message Protocol", STD 5, RFC 792, USC/Information Sciences Institute, September 1981.

## Problem 2. Password of Cabinet

(Time Limit: 2 seconds)

### Problem Description

Karen has a cabinet with a lock of four-digits to store his top secrets. Each digit is one of “0123456789”. In each move, one can move a digit to an adjacent digit in a circular way. For example, one can change “2016” to “2026”, “2006”, or “2916”. It has been a long time since he opened it last time, and he forgets the password.

To recover the password, he doesn’t want to just try all possible passwords. He knows that he never uses a password with duplicated digit like “0000”, “5566”, or “1088” since he thinks these passwords are too weak and not random enough. But there are still 5040 possible four-digits passwords under this condition.

To further speed up the password cracking process, he wants to infer some most possible passwords by the current state of the lock. He thinks he didn’t change the digits of lock significantly after closing the cabinet, and the ones that require minimum number of moves are most possible passwords. For example, if the lock shows “5566” currently, then “4567” is one of the most possible passwords. Please help him to find the most possible passwords.

### Technical Specification

- The number of test case  $T \leq 20000$

### Input Format

The first line contains an integer  $T$  indicating the number of the test cases. For each test case, there is a string with four digits in one line, indicating the current state of lock.

### Output Format

For each test case, please output the number of the most possible passwords, the minimum moves, and the minimum most possible password in one line.

### Example

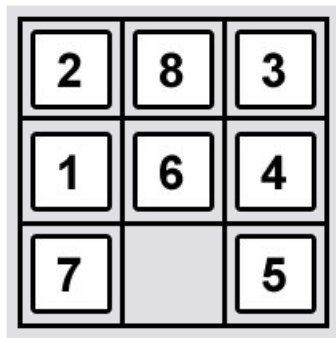
Sample Input:	Sample Output:
3 1234 5566 9999	1 0 1234 4 2 4567 48 4 0189

## Problem 3. 8-puzzle

(Time Limit: 3 seconds)

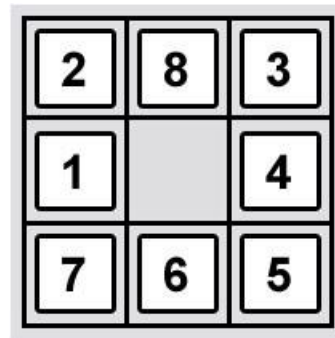
### Problem Description

The 8-puzzle is a 3 x 3 square board with 9 positions, filled by 8 numbered tiles (numbered from 1 to 8) and one empty square. At any step, a tile adjacent to the empty square can be moved into the empty square for changing a new empty square position. For example, Fig. 1 shows an initial board and Fig. 2 shows the result board after making the move of sliding the tile 6 into the empty square.



2	8	3
1	6	4
7		5

Fig.1. An initial board.



2	8	3
1		4
7	6	5

Fig. 2. The result board of sliding tile 6.

The objective of the game is to begin with an arbitrary configuration of tiles (called “initial board”), and move them so as to achieve a specified goal configuration (called “goal board”). In this problem, we are interested in the minimum number of moves from an initial board to the goal board.

### Technical Specification

- A board configuration is represented by a 3 x 3 matrix.
- In the matrix, all tiles are numbered from 1 to 8 which are distinct each other, and the empty square is numbered by 0.

### Input Format

The first line of the input data contains an integer  $n$  indicating the number of test cases. The next three lines, each line contains three integers separated by a space, represent the goal board. In the following, there are  $n$  test cases. In each test case, there are three lines representing the initial board, each line contains three integers separated by a space.

## Output Format

For each test case, if there exists any solution, please output the minimum number of moves from the initial board to the goal board in a line; otherwise output “Infinite” in a line.

## Example

Sample Input:	Sample Output:
3	2
1 2 3	22
4 5 6	Infinite
7 8 0	
1 2 3	
4 5 6	
0 7 8	
0 1 2	
3 4 5	
6 7 8	
1 2 3	
8 0 4	
7 6 5	

## Problem 4. Dragon Slayer

(Time Limit: 5 seconds)

### Problem Description

The Nephalem is on a quest to slay a fire breathing dragon. She needs to acquire 5 essential equipment including a helmet, an armor, a pair of gloves, a shield and a pair of boots in order to defend herself from the flame strike of the dragon. There would be at most 4096 choices for each kind of equipment, which all weigh less than 16384 grams. On the way to the lair of the dragon, there is a bridge that was built many years ago. If the total weight of the equipment is too heavy, the bridge would break down, and the quest would fail. However, if the total weight is too light, the flame strike from the dragon would blow the Nephalem away! If so, the quest would also fail. Please calculate how many ways there are to acquire the equipment so that the Nephalem have a helmet, an armor, a pair of gloves, a shield and a pair of boots that can lead to the success of the quest.

### Technical Specification

- A positive integer  $T$ ,  $1 \leq T \leq 20$
- Two positive integers  $L, U$ ,  $5 \leq L \leq U \leq 81920$
- Five positive integers  $h, a, g, s, b$ , where  $0 < h, a, g, s, b \leq 4096$
- Every equipment has weight less than 16384 grams.

### Input Format

The first line of the input contains an integer  $T$  indicating the number of test data. The first line of each test case contains two integers  $L, U$ , indicating that the flame strike would blow away the Nephalem if her equipment weighs less than  $L$  grams and that she would break the bridge if it weighs more than  $U$  grams. Following is a second line containing 5 integers  $h, a, g, s, b$  separated by spaces representing there will be  $h$  kinds of helmets,  $a$  kinds of armors,  $g$  kinds of gloves,  $s$  kinds of shields and  $b$  kinds of boots. After that are five lines containing  $h, a, g, s$ , and  $b$  integers respectively, corresponding to the  $h$  kinds of helmets,  $a$  kinds of armor,  $g$  kinds of gloves,  $s$  kinds of shields and  $b$  kinds of boots, which are all less than 16384 grams.

## Output Format

For each test case, please output the number representing how many ways there are to slay the dragon with a helmet, an armor, a pair of gloves, a shield and a pair of boots.

## Example

Sample Input:	Sample Output:
2	2
5 5	720
1 2 3 4 5	
1	
1 2	
1 2 3	
1 2 3 2	
1 2 3 2 1	
5 10	
2 3 4 5 6	
1 2	
1 2 1	
1 2 1 2	
1 2 1 2 1	
1 2 1 2 1 2	

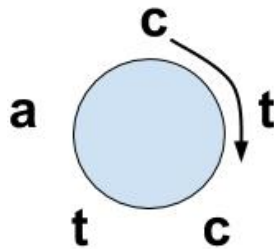


## Problem 5. Pharaoh's Circular Puzzle

(Time Limit: 3 seconds)

### Problem Description

In an old Egypt pyramid, the Pharaoh stored his precious treasure deep under the ground. A long tunnel was built to access to the treasure. The entrance of the tunnel to the treasure is blocked by a huge stone wall. A long encrypted code is written on the stone wall, which was mysteriously generated from an unknown circular string. The ghost of Pharaoh will show up and tell any treasure hunter a spell word. In order to open the stone wall, the hunter must repeatedly speak out the spell word and the repeating frequency must exactly match the clockwise frequency of the spell word in the original circular string. Otherwise, the tunnel to the treasure will collapse. For instance, suppose the original circular string is shown in the following figure. If Pharaoh give a spell word "ct", the hunter should repeatedly saying the spell word twice, because the substring "ct" appears clockwise and twice in the original circular string. If the spell word is "ac", the hunter only need to say once. Note that the spell word may not exist clockwise in the circular string (e.g., ca), and the hunter should keep silence.



However, an encrypted code instead of the original circular string is shown on the wall. Luckily, the treasure hunter knows the secrets of encryption passed from the Pharaoh to his descendants. Both the encrypted code and original circular string are from standard alphabet {a-z}. In order to hide the spell word in the original circular string, all clockwise rotations of this circular string are generated. For example, given a circular string shown in the above figure (ctcta), all possible clockwise rotated sequences at the same length are {ctcta}, {tctac}, {ctact}, {tactc}, and {actct}. Subsequently, all these rotated sequences are sorted lexicographically, leading to {actct}, {ctact}, {ctcta}, {tactc}, and {tctac}. The last letters in these sorted sequences, i.e., {ttacc} forms the encrypted code shown on the stone wall.

Given the encrypted code (e.g., ttacc) and a spell word (e.g., ct), write a program that helps the treasure hunter to answer the clockwise frequency of the spell word in the

original circular string (e.g., 2). Note that the frequency may be zero.

### Technical Specification

- The letters in the original/encrypted strings and spell word are from standard alphabet {a-z}.
- The lengths of original and encrypted strings  $L$  range from 1 to 30000.
- The length of spell word  $W$  ranges from 1 to 30000 and  $W \leq L$ .

### Input Format

The first line contains the number of test cases. For each following two lines, the first line stores encrypted string on the stone wall. The second line stores the string of the spell word.

### Output Format

The output should consist of one line for each test case. Each line contains the clockwise frequency of opening word in the encrypted string.

### Example

Sample Input:	Sample Output:
2 ttacc ct pssmipissii imi	2 1