Reading input is a very important and easy but boring part of programming. So functions which can take difficult sort of inputs very easily are very popular among programmers. Such a function is the ancient scanf() function of C, which exists even today with all its popularity and glory.

The main strength of scanf() function is its format strings. For example "%$d$" is used to read decimal integers, "%$s$" is used for reading strings. One less known but useful format string of scanf() function is to specify the allowed characters within a third bracket pair. For example the following scanf() function will only read decimal digits and will stop reading if it gets anything else:

```
#include<stdio.h>
char s[101];
int main(void)
{
    scanf("%[0123456789]",&s);
    printf("%s\n",s);
}
```

For example, if the input string is "123abc345" then the value of $s$ will be "123", because after reading "123" the scanf() function finds character "a" which is not a decimal digit (Not specified in the format string). So the output shown would be "123".

Given the input string and the format string (In the example above the format string is "0123456789" and input string is "123abc345") it is very easy to determine the output string "123". But in this problem you will have to do the opposite. You will be given a set of input strings and the corresponding output strings, and you will have to determine the format string which will ensure all these inputs and corresponding outputs.

## Input

The input file contains at most 3000 test cases. The description of each test case is given below:

Each test case starts with an integer $N$ ($0 < N < 101$) which denotes how many input/output pairs will follow. The following $N$ lines will contain two strings separated by a single space. The strings are enclosed in double quotation. The first string denotes the input string and the second string denotes the printed string (the characters that were actually read). You can assume that the input and output strings will only contain alpha-numerals ('0'-'9', 'A'- 'Z' and 'a'-'z') and their length will not exceed 100 (excluding the two double quotations) and the input string (first string) will never be an empty string but the output string may be empty. All the strings in the input file are enclosed in double quotation. So an empty string is denoted as "".

Input is terminated by a line containing a single zero.

## Output

For each set of input produce one line of output. This line contains the serial of output followed by the desired format string that will ensure all the output string pairs. Make sure that this format string contains only alpha-numerals. If there is more than one possible format string, output the ASCII-graphically smallest one. (Lexicographically means word ordering as if they were in lexicon or dictionary. So "and" is smaller than "bbt" because "and" would have appeared before "bbt" if they both were kept in a dictionary. ASCII-cographically sorted means sorted in the order in a lexicon where all the ascii characters are the alphabets. So here "abc" is smaller than "cde", "Mccain" is smaller than "barak" etc.)

The format string must contain at least one alpha-numeral character and should contain no alpha-numeral more than once. If no format string of positive length and consisting only alpha-numerals can produce such output or the input output pair seems impossible then print the line 'I_AM_UNDONE' instead (without the quotes). Note that the format string is printed enclosed in a third bracket. Look at the output for sample input for formatting details.

## Sample Input

```
5
"11" "11"
"243" "24"
"563" "56"
"784" "784"
"789" "78"
1
"A" "b"
0
```

## Sample Output

```
Case 1: [01245678]
Case 2: I_AM_UNDONE
```