A. Search For Patterns!

Score: 1  
  
CPU: 1s  
Memory: 1024MB

Searching for patterns is a very attractive field. Who didn’t wish to discover the patterns of Grameen Phone recharge cards!

In this problem, first, you have to know how patterns can be subsequence of a given string. Suppose S and P are two strings. Here P will be subsequence of S if P can be derived from S by deleting some elements without changing the order of the remaining elements.

For example,

S = BLEALBIE

|| | |

P = BL A I

So, P(BLAI) is a subsequence of S(BLEALBIE).

We define the “First Lookup Subsequence” as follows:

For each character, c[i] (0 ≤ i < |P|, for a string X, |X| = length of X), in P, we mark the first occurrence of c[i] in S and write down the positions of c[i] in S as, pos[0], pos[1], … pos[|P| - 1], where pos[i] denotes the index in S where c[i] is first located (left to right searching). If these values form an increasing series, that is, pos[0] < pos[1] < pos[2] <….< pos[|P| - 1], then we say that S contains P as a “First Lookup Subsequence”.

In this problem, you will be given two strings, S and P, containing only uppercase letters of English alphabet (A-Z). Each character of S is distinguishable, that is, two ‘A’s are considered different. (You can assume all letters are of different colors! so that they are distinguishable). Each character in P is distinct. Your job is to find how many permutations of S contain P as a First Lookup Subsequence. Be careful about the permutations of S. Although two strings might look same, they can be of different permutations.

For example, for a string, S = AAE, we assume 3 different colors.

A(red) A(blue) E(purple)

So it has 6 different permutations

A(red) A(blue) E(purple) => AAE

A(red) E(purple) A(blue) => AEA

A(blue) A(red) E(purple) => AAE

A(blue) E(purple) A(red) => AEA

E(purple) A(red) A(blue) => EAA

E(purple) A(blue) A(red) => EAA

If we search the pattern P(AE), as a First Lookup Subsequence in all these permutations, permutation 1,2,3,4 will contain P as a First Lookup Subsequence.

So the number of permutations of S, that contain P as a First Lookup Subsequence, is 4.

Input

The first line of input contains a single integer, T (T ≤ 100), denoting the number of test cases to process. Next, there are T test cases. Each contains two strings S and P in separate lines. Here, 0 < |S| ≤ 500, 0 < |P| ≤ 26. All the letters in S and P will be uppercase English letters (A-Z). All the letters in P will be distinct.

Output

For each case, print a line of output in the following format:

Case n: m

Where n is the test case number and m is the output modulo 100007.

Sample

| **Input** | **Output** |
| --- | --- |
| 5  AAE  AE  AADE  DE  AADEBG  GDA  A  A  EEEEEE  E | Case 1: 4  Case 2: 12  Case 3: 60  Case 4: 1  Case 5: 720 |