# Problem 4 - Too Many Emails (100 pts)

| | | |
|---|---|---|
| Time Limit | : | 3 s |
| Memory Limit | : | 1024 MB |

## Problem Description

Lingling recently receive her Ph.D. degree and became a professor that has always been her dream job. But some chores bother her. There are too many emails sent to her every day! Also, the mail system that she uses is very old. The system sometimes adds garbled text into her emails, and has very little storage space.

To solve the first problem, Lingling makes some observations and found that garbled text exhibits some patterns:

1. It's a substring of an email, and each email only has at most one garbled text.
2. Garbled text is known to have some specific characters. Some might occur for multiple times.

Lingling comes up with an idea that can locate garbled text in an email. First, based on recent experience, she writes down a string of garbled text characters, denoted $G$. In $G$, assume we have $n$ different characters, $c_1, c_2, \ldots, c_n$, with the number of occurrences in $G$ denoted as $o_1, o_2, \ldots, o_n$, respectively. For example, if $G$="TGET", then we have $c_1$='T', $c_2$='G', $c_3$='E', and $o_1 = 2, o_2 = 1, o_3 = 1$. To locate the garbled text within the email text $D$, we look for a substring $D[i..j], 1 \leq i, j \leq |D|$, such that the number of occurrences of $c_1, c_2, \ldots, c_n$ within $D[i..j]$, denoted $p_1, p_2, \ldots, p_n$, can satisfy the inequalities $p_1 \geq o_1, p_2 \geq o_2, \ldots, p_n \geq o_n$. Note that we want to find a minimum length garbled text $D[i..j]$ satisfying the above conditions, in order to retain the most email content. Then $D[i..j]$, identified as garbled text in the email, can be removed from $D$, obtaining the email without garbled text, denoted as $D'$.

For example, given email text $D$ ="DSA is so GaRBledTExTeasy:D" and garbled text string $G$ ="TGET". To see if substring $D[11, 21]$ ="GaRBledTExT" is a garbled text, we evaluate the number of occurrences of $c_1$='T', $c_2$='G', $c_3$='E' within $D[11, 21]$, and obtain $p_1 = 2, p_2 = 1, p_3 = 1$. Since they satisfy $p_1 \geq o_1, p_2 \geq o_2, p_3 \geq o_3$, $D[11, 21]$ is considered garbled text. Moreover, we cannot find any other substring of $D$ satisfying these conditions and has a length smaller than $D[11, 21]$. Finally, we remove $D[11, 21]$ from the original $D$, obtaining the final email text $D'$ ="DSA is so easy:D".

The next step is to save the space to store the email. The main idea is to break $D'$ into $k$ substrings, called blocks, denoted as $b_1, b_2, \ldots, b_k$. Each block is a substring of $D'$, i.e., $b_i = D'[s_i..s_{i+1} - 1], 1 \leq s_i \leq |D'|, s_1 = 1, s_{k+1} = |D'| + 1$, but can have different lengths. Most importantly, we have $b_i == b_{k-i+1}$, for $1 \leq i \leq k$. In this case, the email system only needs to store $b_i$ but does not need to store $b_{k-i+1}$, for $1 \leq i \leq \lfloor k/2 \rfloor$. This can effectively save up to 50% of the length of the email text. Note that the system prefers breaking the email into small blocks. Therefore, you are requested to break the email into **as many blocks as possible**.

For example, $D'$="ABDCDAB" can be broken into $b_1$ ="AB",$b_2$ ="D",$b_3$ ="C",$b_4$ ="D", $b_5$ ="AB", saving space to store $b_4$ and $b_5$.

Can you write a program to help Lingling remove garbled text from the email, and then find a way to break the email into $k$ blocks to save the storage space?

## Input

The first line contains an integer T, indicating the number of test cases. For each test case, there are two lines, which contain string $D$ and string $G$ respectively, representing the email text and the garbled text string. $D$ and $G$ only contain English characters (upper- and lower-case letters should be treated as different characters). If there are multiple garbled texts of the same length, remove the left most one.

## Output

For each test case, print string $D'$ with garbled text removed and using '|' to indicate where to break $D'$ into blocks. Things that need to be pay attention to include:

1. There are cases where the garbled text conditions are not satisfied with any of the substrings of $D$. In this case, you do not need to delete any character from $D$ to obtain $D'$.

2. There are cases where $D$ cannot be broken into blocks which satisfied the block conditions. In this case, you do not need to add any '|' to $D'$.

3. You can assume that length of the output is always larger than or equal to 1.

## Constraints

1. $1 \leq T \leq 10^2$

2. $1 \leq |G| \leq |D| \leq 10^5$

**Subtask 1 (20 pts)**

- $1 \leq |G| \leq |D| \leq 10^3$

**Subtask 2 (15 pts)**

- D does not contain any character in G.

**Subtask 3 (15 pts)**

- D cannot be broken into more than one blocks. (No '|' in the output)

**Subtask 4 (50 pts)**

- No other constraints.

## Sample Input 1

```
1
HelloWorldolleH
ee
```

## Sample Output 1

```
H|H
```

## Sample Input 2

```
2
DSARANDOMTEXTISSOHARD
RTTX
DSARANDOMTEXTISSOEASY
RXTT
```

## Sample Output 2

```
D|SAISSOHAR|D
DSAISSOEASY
```

## Sample Input 3

```
3
OkayOkayOkay
x
Nooo
oo
JJJJJJ
j
```

## Sample Output 3

```
Okay|Okay|Okay
No
J|J|J|J|J|J
```