G. Merge the Strings

Score: 1

CPU: 1s
Memory: 1024MB

You are given two strings **A** and **B** which you are required to merge into a single string **C**. To merge **A** and **B**, we need to find some of their common non-overlapping substrings. After that, we need to group those common substrings and the remaining uncommon substrings separately and merge these groups to form **C**. The merging needs to be done in such a way that the order of the characters of **A** and **B** will remain the same in **C**. This means that both **A** and **B** will be subsequences of **C** and it will not contain any extra characters which aren't present in either **A** or **B**. We can indicate the common substrings, the substrings which only originate from **A** and the substrings which only originate from **B** by surrounding them with **()**, **{}** and **[]** respectively.

For example, if **A = BABBA** and **B = BBBAC** one possible way to get common non-overlapping substrings would be **(B)** and **(A)**, so in their grouped form **A = (B)(A){BBA}** and **B = (B)[BB](A)[C]**. A possible merge can be **C = BBBACBBA** with its grouped form being **(B)[BB](A)[C]{BBA}**.

Another possible way to get common non-overlapping substrings would be **(B)** and **(BBA)**, so in their grouped form **A = (B){A}(BBA)** and **B = (B)(BBA)[C]**. The only possible merge for this is **C = BABBAC** with its grouped form being **(B){A}(BBA)[C]**.

Note that even though **A** and **B** were the same, two different **C**s were obtained with the latter one having smaller length. Your task is to find such a **C** which has the smallest length and group the characters so that you can tell which character originates from which string. You need to surround a substring of **C** with **()**, **{}** or **[]** if it originates from both **A** and **B**, only **A** or only **B** respectively.

If you obtain multiple **C** of the smallest size, find the one which is lexicographically smallest after grouping the characters. For this problem, assume that **'A' < ... < 'Z' < '(' < ')' < '{' < '}' < '[' < ']'**.

For example, if **A = BABBA** and **B = BBBAC**, you will obtain **C = BABBAC** which has the minimal length 6 and will have the grouped form of **(B){A}(BBA)[C]**.

Note:
A subsequence **R** of a string **S** is a sequence which is obtained by removing zero or more characters from **S**. A substring **R** of a string **S** is a continuous block of characters of **S** in the inclusive range from index **l** to **r**, where **R =** $\mathbf{S_lS_{(l+1)}S_{(l+2)}...S_r}$ and **1 <= l <= r <= |S|**.

Input

The first line of the input contains a single integer **T**, which denotes the number of test cases. This is followed by the test cases. Each case consists of two lines which contain **A** and **B** respectively.

Constraints
1 <= T <= 30
1 <= |A|,|B| <= 50
All characters are uppercase letters.

Output

For each test case, output the case number, followed by the length of C and it's grouped form, each in it's own separate line. Again, if you obtain multiple C of the smallest size, output the grouped form which is lexicographically smallest. For this problem, assume that 'A' < … < 'Z' < '(' < ')' < '{' < '}' < '[' < ']'.

See the sample input/output for more clarification.

Sample

Input

```
3
BABBA
BBBAC
ABC
ABC
ABC
XYZ
```

Output

```
Case 1:
6
(B){A}(BBA)[C]
Case 2:
3
(ABC)
Case 3:
6
{ABC}[XYZ]
```