Security

ou are designing a new encryption system that works in the following way:

For server-client communication you need a key **k**, composed of **m** sections, each of length **l**, and the key consists only of lowercase characters in the set {a, b, c, d, e, f}. The server has a key **k1** and the client has a key **k2** where:

k1 = f(k). **f** is a function that receives a key and replace some random letters by ? indicating that those characters can be any lowercase letter of the set described before.
k2 = f(g(k)). **g** is a function that takes a key and produces a random permutation of its m sections. And **f** is the function defined above.

For example: let m = 3, l = 2

f('abacbc') = '?ba??c'
g('abacbc') = 'acbcab' (each section was moved one place to the left).

Your task is given **k1** and **k2**, find key **k**. If there are several solutions, print the lexicographically smallest key. And if there is no solution at all, print "IMPOSSIBLE" (without the quotes).

Input

The first line has a single integer **T**, which corresponds to the number of test cases. **T** test cases follows: the first line of the test case corresponds to the integer **m**, the second line contains the string **k1** and the third line contains the string **k2**.

Output

For test case **i**, numbered from **1** to **T**, output "Case #i: ", followed by the lexicographically smallest key or "IMPOSSIBLE".

Example

Input:

5

2

abcd

c?ab

3

ab?c?c

ac?c??

3

ab?c?c

aabbdd

2

aa

bb

2

abcd

cdab

Output:

Case #1: abcd

Case #2: abacac

Case #3: IMPOSSIBLE

Case #4: IMPOSSIBLE

Case #5: abcd

Constraints :

T <= 20

0 < |k1| <= 100

0 < m <= 50

|k2| = |k1|

It is guaranteed that m is always a divisor of |k1|

k1 and k2 consist of {a, b, c, d, e, f, ?}