Some students of Computer Science at UFPE decided to create a project based on the Peer-to-Peer model to transfer data. They also decided to implement an unique method of cryptography based on symmetric-keys algorithms so that any messages could be encrypted and decrypted using the same key. They called this method "cryptography of the floating key". It receives as input a string P of lowercase letters and a key (N, M), and returns the encrypted message Q. We'll now describe the steps to encrypt a string P with a key (N, M).

1. If $N * M \geq |P|$, go to step 2. Otherwise, if N and M are different, take the smallest of them and increment it with the value of the largest of them; if they are equal, double the value of M. Return to step 1.

2. Create a $N \times M$ matrix and arrange the string P in this matrix: the first line of the matrix should be occupied by the first M characters of P, the second line of the matrix should be occupied by the following M characters of P, and so on. There'll probably be some empty cells on the matrix by the end, but you should not worry about it.

3. Sweep each element of the matrix that is not an empty cell and check its position. The rows and columns are indexed starting from 1. If an element is on the position (i, j) of the matrix, than make a shift of $i + j$ on the character. That is, if there's a character 'a' on the position $(1,1)$ of the matrix, it turns to 'c'; if there's a character 'z' on the position $(1,3)$ of the matrix, it turns to 'd'.

4. From the first row and first column, swap adjacent elements of the matrix in a zigzagging way. That is, given that you are in position $(1,1)$, you'll swap the element at $(1,1)$ with the element at $(1,2)$. Then, go to the right. Now, you are in position $(1,2)$, but the element at $(1,2)$ has already been swapped, so you don't swap it and keep going right. When you get to position $(1, M)$, if it has been swapped, just go down to position $(2, M)$. If it hasn't, swap it with the element $(2, M)$ and go down. Now that you are at $(2, M)$, you'll start going left, applying the same algorithm, and when you get to $(2,1)$, you'll go down and start going right. The line of the matrix with the last part of P should be left intact, so you can't swap any elements in that line. If you come to a situation where you have to swap an element with the element below it, but the line below is the line containing the last part of the string P, then you stop immediately.

5. Now you should rotate the matrix 90 degrees to the left, and the string should be read from left to right, top to bottom. Empty spaces of the matrix should be ignored. As result, you'll obtain the encrypted string Q.

P = "maratonaehlegal"
N = 3
M = 3

Q = "zsvmvjfnroqfdek"
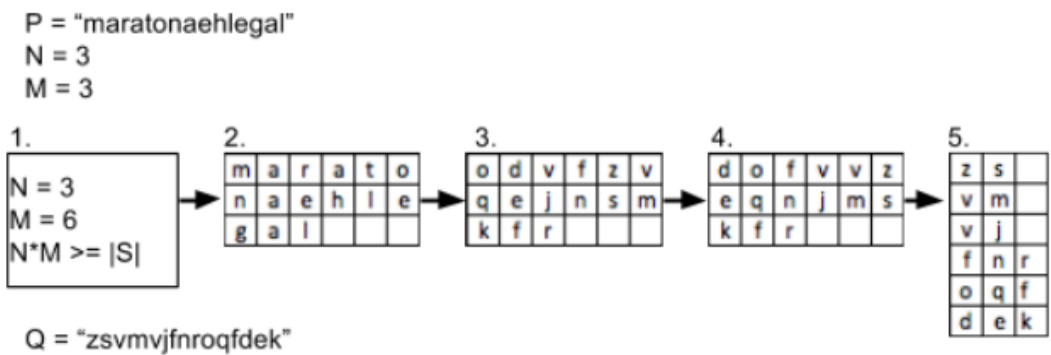
Somehow, the piece of code responsible of decrypting the messages was infected by a jealous virus, and they called you to create a new program. You must solve the following problem: given the encrypted message Q and the key (N, M), decrypt the message and return the initial string P.

Input

The first line contains T $(T \leq 10^4)$ — the number of test cases, after this line T test cases follows. Each test case is arranged in a line containing a string Q $(1 \leq |Q| \leq 100)$ and two integers N and M $(1 \leq N, M \leq 100)$ — the encrypted message and the two values that make up the key, correspondingly.

Output

For each test case print a line containing 'Case #X: Y', where X is the case number, starting at 1, and Y is the initial string P.

Sample Input

```
3
vnvxjbxylndjhzq 1 1
zsvmvjfnroqfdek 3 3
uhyeobnqcmayftjvtttrplnraykamqw 50 100
```

Sample Output

```
Case #1: letshavesomefun
Case #2: maratonaehlegal
Case #3: universidadefederaldepernambuco
```