Layered Caesar Salad

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 mebibytes

"What solution did you have?"
"Brute."
"Me too, brute."
"*Et tu, brute?*"

From a contestants' discussion

Behold an innovative encryption method, inspired by the balance of ancient wisdom and modern simplicity!

You must be familiar with the Caesar cipher. It is exceptionally simple: each letter of the message is shifted right in the alphabet by the same number of positions. If shifting goes beyond the alphabet, it proceeds from its beginning. For example, the word `fusion` with a shift of 6 becomes the word `layout`. To decrypt the message, each letter must be shifted left by the same number.

The new encryption method is called the *Caesar Salad cipher*. It is very similar to its ancient prototype but is more reliable! It is applied to words consisting of letters from `a` to `z`. Each letter of the Latin alphabet corresponds to a numerical code from 0 to 25: the letter `a` has code 0, the letter `b` has code 1, `c` has code 2, and so on up to the letter `z` that has code 25. The Caesar Salad cipher works as follows:

1. Let $(a_1, a_2, \ldots, a_k)$ be the numerical codes of the letters of the original message of length k. For example, the word `delta` corresponds to the sequence of codes $(3, 4, 11, 19, 0)$.
2. Choose a shift x, an integer from 1 to 25 inclusive. Note that, just as in the original Caesar cipher, a zero shift cannot be chosen for security reasons.
3. Assign $b_1 := (a_1 + x) \mod 26$.
4. Next, for each i from 2 to k, assign $b_i := (a_i + b_{i-1} + x) \mod 26$.
5. The sequence of codes $(b_1, b_2, \ldots, b_k)$ corresponds to the encrypted message. So, after encrypting the word `delta` with a shift of $x = 20$, we get $(23, 21, 0, 13, 7)$ which corresponds to `xvanh`.

And now, without further ado, it's time to apply your new knowledge in practice! You are to develop a protocol for transmitting a digital key. A digital key consists of n words, each containing k Latin letters. Among these words, there may be duplicates, but in general, their order does not matter. Before transmission over open channels, n new words of length k are mixed in, which are the original words passed through the Caesar Salad cipher. Each of the n words is allowed to be encrypted with different shifts. The resulting multiset of $2n$ words, shuffled in arbitrary order, is called the digital key *with admixture*.

Your task is to implement both parts of the protocol. In the first part, based on the given digital key, you need to generate a key with admixture, and in the second part, extract all the words of the original digital key from the key with admixture generated by your algorithm.

Input

Your program will be run **twice** on each test. In the first run, the input will consist of the original data, and in the second run, there will be the data obtained after your encryption.

Each test contains multiple test cases. The first line contains one word S of uppercase Latin letters and one integer t ($1 \le t \le 1000$): the number of test cases. If $S =$ `ENCODE`, this is the **first run**, and for each test case, you are required to generate a digital key with admixture. If $S =$ `DECODE`, this is the **second**

run, and for each test case, you are required to extract the words of the original digital key from the key with admixture. The description of the test cases follows.

The first line of each test case contains two integers n and k ($1 \le n \le 25$, $1 \le k \le 20$): the number of words in the original digital key and the length of each of these words.

If $S =$ `ENCODE`, each of the next n lines contains one word of k lowercase Latin letters. These n words define the original digital key. Some words may be identical.

If $S =$ `DECODE`, each of the next $2n$ lines contains one word of k lowercase Latin letters. These $2n$ words define the digital key with admixture, output by your program in the first run. The order of words in the key with admixture may be arbitrary.

Output

If $S =$ `ENCODE`, for each test case, output n words of length k: the words encrypted with the Caesar Salad cipher. Different shifts may be chosen for each word, but the output order of the encrypted words must correspond to the order of the original words. These encrypted words will be mixed with the n original words before the second run.

If $S =$ `DECODE`, for each test case, output n words of length k: the restored digital key that must match the original key. You may output the words of the digital key in any order during the second run.

The case of letters matters: the output Latin letters must be lowercase.

Example

standard input	standard output
ENCODE 2 4 5 delta alpha alpha prime 1 20 petrozavodskprogcamp DECODE 2 4 5 alpha prime kfevf delta zjxdc alpha rkuio xvanh 1 20 vfebvaghbkiytqkwekcx petrozavodskprogcamp	xvanh zjxdc kfevf rkuio vfebvaghbkiytqkwekcx alpha delta prime alpha petrozavodskprogcamp

Note

The example illustrates the format of input and output data for both runs. The descriptions of data for the first and second runs are separated by a blank line for clarity.

For the first test case, shifts of 20, 25, 10, and 2 were chosen for the words, respectively.

For the second test case, after the first run, the word was encrypted with a shift of 6.