Permutation Code

As the owner of a computer forensics company, you have just been given the following note by a new client:

I, Albert Charles Montgomery, have just discovered the most amazing cypher for encrypting messages. Let me tell you about it.

To begin, you will need to decide on a set of symbols, call it *S*, perhaps with the letters RATE. The size of this set must be a power of 2 and the order of the symbols in *S* is important. You must note that R is at position 0, A at 1, T at 2, and E at 3. You will also need one permutation *P* of all those symbols, say TEAR. Finally you will need an integer, call it *x*. Together, these make up the key. Given a key, you are now ready to convert a plaintext message *M* of length *n* (*M*[0], *M*[1]... *M*[*n*-1]), that has some but not necessarily all of the symbols in *S*, into a cyphertext string *C*, also of length *n* (*C*[0], *C*[1],...*C*[*n*-1]), that has some but not necessarily all of the symbols in *S*.

The encrypting algorithm computes *C* as follows:

1. Calculate an integer *d* as the remainder after dividing the integer part of ($n^{1.5} + x$) by *n*. This can be expressed more succinctly as $d = (\text{int})(n^{1.5} + x) \% n$, where "%" is the remainder operator.
2. Set *C*[*d*] to be the symbol in *S* whose position is the same as the position of *M*[*d*] in *P*.
3. For each $j \neq d$ in 0..*n*-1, set *C*[*j*] to be the symbol in *S* whose position is the value obtained by xor-ing the position of *M*[*j*] in *P* with the position of *M*[(*j*+1) % *n*] in *S*. Note that the bitwise xor operator is "^" in C, C++, and Java.

For example, consider this scenario where *S*=RATE, *P*=TEAR, *x*=102, *M*=TEETER, and *n*=6. To compute *d*, first calculate $6^{1.5} + 102 = 116.696938$, then take the remainder after dividing by 6. So *d* = 116 % 6 = 2. The following table shows the steps in filling in the cyphertext *C*. Note that the order of the steps is not important.

	0	1	2	3	4	5	
S =	R	A	T	E			
P =	T	E	A	R			
M =	T	E	E	T	E	R	
C =	E						*M*[0] is T, T is at *P*[0]. *M*[1] is E, E is at *S*[3]. *C*[0] = *S*[0 xor 3] = *S*[3]
	E	T					*M*[1] is E, E is at *P*[1]. *M*[2] is E, E is at *S*[3]. *C*[1] = *S*[1 xor 3] = *S*[2]
	E	T	A				*2 is d. M*[2] is E, E is at *P*[1], so *C*[2] = *S*[1]
	E	T	A	E			*M*[3] is T, T is at *P*[0]. *M*[4] is E, E is at *S*[3]. *C*[3] = *S*[0 xor 3] = *S*[3]
	E	T	A	E	A		*M*[4] is E, E is at *P*[1]. *M*[5] is R, R is at *S*[0]. *C*[4] = *S*[1 xor 0] = *S*[1]
	E	T	A	E	A	A	*M*[5] is R, R is at *P*[3]. *M*[0] is T, T is at *S*[2]. *C*[5] = *S*[3 xor 2] = *S*[1]

I have included additional examples of encrypted messages at the end of this note for you to experiment with. However, first, I need to tell you about the decryption algorithm.

Unfortunately, the next page of the note, with the decrypting algorithm, is completely unreadable because it is covered with huge, overlapping, messy ink blots. Given your considerable skill in unravelling puzzles, your task is to write the decoder based on your knowledge of the encoding algorithm.

Input

The input for the decoder consists of one or more sets of {key, encrypted message} pairs. The key is on 3 separate lines. The first line contains the single integer *x*, $0 < x < 10{,}000$; the second line contains the string *S*;

and the third line contains the string *P*, which will be a permutation of *S*. The length of *S* (and therefore *P*) will always be one of the following powers of two: 2, 4, 8, 16, or 32. Following the key is a line containing the encrypted message string *C*, which will contain at least one and at most sixty characters. The strings *S*, *P*, and *C* will not contain whitespace, but may contain printable characters other than letters and digits. The end of the input is a line which contains the single integer 0.

Output

For each input set print the decrypted string on a single line, as shown in the sample output.

Example

Input:

```
102
RATE
TEAR
ETAEAA
32
ABCDEFGHIJKLMNOPQRSTUVWXYZ._!?,;
;ABCDEFGHIJKLMNOPQRSTUVWXYZ._!?,
MOMCUKZ,ZPD
1956
ACEHINT_
ACTN_IHE
CIANCTNAAIECIA_TAI
0
```

Output:

```
TEETER
HELLO_WORLD
THE_CAT_IN_THE_HAT
```