Problem F
Leapfrog Encryption
Time Limit: 1 second, Memory limit: 2G

We've come up with a new encryption method we call Leapfrog Encryption. It is a key-based encryption scheme where an alphabetic key specifies how letters in the *plaintext* (the text to be encrypted) are placed in the *ciphertext* (the resulting encrypted string). Here's how Leapfrog Encryption works:

1. Remove all non-alphabetic characters from the plaintext and convert all remaining letters to lowercase.
2. Convert the letters of the key to their location in the alphabet $+1$ (so that '`a`' converts to 2, '`b`' converts to 3 and so on). This gives us a sequence of numbers $d_1, d_2, \ldots, d_n$ where n is the length of the key.
3. Going left-to-right, place the first letters in the plaintext in every d_1-th location of the ciphertext until you run out of positions in the ciphertext (where the length of the ciphertext is equal to the number of letters in the plaintext). So for example, if $d_1 = 5$ the first letter in the plaintext goes in position 5 of the ciphertext (numbering the first location in the ciphertext as position 1), the second letter in the plaintext goes in position 10 in the ciphertext, and so on.
4. Repeat this with d_2 but this time going right-to-left through the ciphertext, only counting the empty positions (leapfrogging over the letters already in the ciphertext).
5. Continue with d_3, d_4, etc., alternating the direction you go through the ciphertext each time.
6. If there are still letters left in the plaintext after using d_n, fill in the remaining empty locations in the ciphertext with these remaining letters, again going in the opposite direction of the previous pass (this is equivalent to having $d_{n+1} = 1$).

For example, if our plaintext is "Send more monkeys!" and our key is "bea", the encryption proceeds as follows:

b $\rightarrow$ 3, left-to-right:	`_ _ s _ _ e _ _ n _ _ d _ _ m`
e $\rightarrow$ 6, right-to-left:	`_ _ s _ _ e o _ n _ _ d _ _ m`
a $\rightarrow$ 2, left-to-right:	`_ r s _ e e o _ n m _ d o _ m`
last pass, right-to-left:	`s r s y e e o e n m k d o n m`

Decryption is done by . . . hey, you know what? We're going to let you figure that out.

Input

The first line of input contains two strings t k where t is either `E` or `D` indicating whether to perform encryption or decryption and k is the lowercase alphabetic key. The length of k will be between 1 and 100, inclusive. The second and final line of input contains the plaintext to encrypt (if t is `E`) or the ciphertext to decrypt (if t is `D`). This string is non-empty and has a maximum length of $2\,000$. A ciphertext string consists of lowercase letters only, while a plaintext string may contain uppercase letters, numbers, punctuation and spaces as well (all counted as part of the length of the string) and is guaranteed to contain at least one letter.

Output

Output the encrypted or decrypted text. Your output should only contain lowercase letters.

Sample Input 1

```
E bea
Send more monkeys!
```

Sample Output 1

```
srsyeeoenmkdonm
```

Sample Input 2

```
D bea
srsyeeoenmkdonm
```

Sample Output 2

```
sendmoremonkeys
```

Sample Input 3

```
D zyxwvutsrqponmlkjihgfedcba
lafogrpe
```

Sample Output 3

```
leapfrog
```