Project Euler #59: XOR decryption

This problem is a programming version of Problem 59 from projecteuler.net

Each character on a computer is assigned a unique code and the preferred standard is ASCII (American Standard Code for Information Interchange). For example, uppercase A = 65, asterisk $(*)$ = 42, and lowercase k = 107.

A modern encryption method is to take a text file, convert the bytes to ASCII, then XOR each byte with a given value, taken from a secret key. The advantage with the XOR function is that using the same encryption key on the cipher text, restores the plain text; for example, 65 XOR 42 = 107, then 107 XOR 42 = 65.

For unbreakable encryption, the key is the same length as the plain text message, and the key is made up of random bytes. The user would keep the encrypted message and the encryption key in different locations, and without both "halves", it is impossible to decrypt the message.

Unfortunately, this method is impractical for most users, so the modified method is to use a password as a key. If the password is shorter than the message, which is likely, the key is repeated cyclically throughout the message. The balance for this method is using a sufficiently long password key for security, but short enough to be memorable.

Your task has been made easy, as the encryption key consists of three lower case characters (a-z). Using the knowledge that the plain text must contain common English characters (a-z, A-Z, 0 - 9), brackets (), common symbols (;:,.'?-!) and spaces decrypt the message and find the key that is used to encrypt the message.

Note It is guaranteed that key is unique, is of size 3 and contains lower case english characters (a-z).

Input Format
First line of input contains an integer N. Followed by N space separated integers denoting the encoded ASCII codes.

Output Format
Print the answer corresponding to the test case.

Constraints
$80 \le N \le 1500$

Sample Input

```
82
32 66 50 20 11 0 42 66 33 19 13 20 47 66 37 14 58 67 43 23 14 17 49 67 46 20 6 51 66 55 9 39 67 45 3 25 56 66 39 14
37 34 65 51 22 8 1 40 65 32 17 14 21 45 65 36 12 57 66 41 20 15 19 50 66 44 23 7 49 65 54 11 36 66 47 0 24 58 65 38
12 38
```

Sample Output

```
abc
```