Problem 3-E4 – Text encryption

Caesar's cipher is one of the simplest methods for encrypting messages: each letter in the message is replaced by a letter that is in the alphabet at a fixed ("key") distance from the replaced one. More keys can be used to obtain a more efficient encryption.

Requirement

Given a certain text (string), the number of encryption keys and the values of these keys (as integers), perform a multi-key encryption as following. The first letter of the text is modified using the first key. This is done by replacing the letter with the alphabet letter which is found at a distance from the current letter equal to the value of the key. The second letter of the text is changed using the second key, based on the same principle, and so on, until all the keys are finished. After all the keys are used, they are to be reused in the same order (circular way), from the beginning, until all the text letters are changed. We use the following alphabet: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z (capital or small letters), with the observation that capital letters will be changed also with capital letters and small letters will be changed with small letters. The alphabet is used also in a circular way. For example, if 3 is the considered key, then 'Z' is replaced with 'C' and 'y' is replaced with 'b'. The other characters from the text that are not part of the alphabet are not encrypted. Display the modified text after the encryption.

Entry data

The program will receive, from the keyboard (*stdin* stream), the following data:

- one text (string) on a single line, followed by *newline* (*Enter* key);
- an integer value ***n***, as the number of encryption keys, followed by *newline* (*Enter* key);
- the integer values of the ***n*** encryption keys, as one value on a single line followed by *newline* (*Enter* key).

Output data

The program will display, on a single line, the text obtained from the encryption, followed by *newline* character (*Enter* key).

WARNING to the compliance of the problem requirement: displaying the results should be EXACTLY on the indicated way! In other words, the standard output stream will not show anything in addition to the requirements of the problem; as a result of automatic evaluation, any additional character displayed, or viewed other than that indicated, will lead to a false result and therefore obtain a Rejection of the program.

Restrictions and specifications

1. The text is entered without diacritics and may contain spaces and punctuation marks that remain as such in the encrypted text (i.e., are not encrypted). The text contains at least 2 characters but no more than 256;
2. The ***n*** value is an integer in the [2; 10] interval;
3. **Warning**: Depending on the chosen programming language, the file containing the code must have one of the following extensions .c, .cpp, .java, or .m. The web editor **will not automatically add an extension** and its absence leads to the impossibility of compiling the program!
4. **Warning**: The source file must be named, by the candidate, in the following form: <name>.<ext>, where name is the surname of the candidate and the extension is chosen according to the previous point. Pay attention to the limitations of the Java language, related to the class name and file name!

Example

Input	**Output**

`ABCZ efgh!`
`3`
`1`
`2`
`3`
	`BDFA gihj!`

Explanation: Input text = “ABCZ efgh!” ‘A’ is replaced with ‘B’ (key 1 – B is at distance 1 of A) ‘B’ is replaced with ‘D’ (key 2 – D is at distance 2 of B) ‘C’ is replaced with ‘F’ (key 3 – F is at distance 3 of C) ‘Z’ is replaced with ‘A’ (key 1 – A is at distance 1 of Z) ‘ ’ remains unchanged ‘e’ is replaced with ‘g’ (key 2 – g is at distance 2 of e) ‘f’ is replaced with ‘i’ (key 3 – i is at distance 3 of f) ‘g’ is replaced with ‘h’ (key 1 – h is at distance 1 of g) ‘h’ is replaced with ‘j’ (key 2 – j is at distance 2 of h) ‘!’ remains unchanged Encrypted text = “BDFA gihj!”	

Working time: 120 minutes