Yet Another Chat App

We need your help in building a chat application for a new type of device called Com-link. Com-link connects to other similar devices with a limited cutting-edge network where we can send digits from 0 to 9 rather than binary 0 and 1. So we have come up with a newer model of encoding as described below.

Example 1:

Given the key:
“r”, “sw”, “x”, “c”, “nm”, “q”, “yt”, “z”, “ab”

If I want to send the message “bat” then the encoding result will be “99090770”

Two 9’s represent the second character in the ninth string, which happens to be “b”. 0 is a delimiter. One 9 represents “a” as the first character in the 9th string is “a”, then 0 as a delimiter. For “t” we had to use two 7’s.

If I want to send the message “goat” then encoding is not possible. As “g” is not available in the key.

If we want to send a message “BAT” then the result will be “09900900770”. Adding an extra “0” at the start of the character’s encoding turns it into uppercase.

Example 2:

Given the key:
“1abc”, “2def”, “3ghi”, “4jkl”, “5mno”, “6pqr”, “7stu”, “8vwx”, “ 90yz”

If I want to send the message “5 Roses” then the encoding result will be “5090066660555507702220770”

One 5 represents the first character in the fifth string, which happens to be “5”. 0 is a delimiter. One 9 represents the space as the first character in the ninth string is a space, then a delimiter. For “R” we had to use 0 for capitalization and four 6’s then a 0 delimiter. Four 5’s represent ‘o’ and so on.

On the receiving end, we have to transfer the encoded result back to the actual string using the same key by reversing the process.

Input:

The input will be read from a file. The first line of the file will be the key. The second line will describe the operation, 1 for encode and 2 for decode. The third line will contain the string to encode or decode.

Output:

The output will be a single line of encoded or decoded results or “Error” if decoding or encoding is not possible.

Constraints:

- The key will have a max of 9 strings.
- Encoding and decoding of a UTF-8 character set is expected.
- 0 <= Length of each string in the key <= 100
- 0 will always be the delimiter.
- 0 at the start will indicate conversion to uppercase.

- 0 < Length of string to encode or decode < 1000000

Sample 1

Input:
“r”, “sw”, “x”, “c”, “nm”, “q”, “yt”, “z”, “ab”
1
bat

Output:
99090770

Sample 2

Input:
“r”, “sw”, “x”, “c”, “nm”, “q”, “yt”, “z”, “ab”
2
099090770

Output:
Bat

Sample 3

Input:
“r”, “sw”, “x”, “g”, “nm”, “q”, “yt”, “z”, “ab”
1
Cat

Output:
Error

COMPLETE

Your Response

Status

Your response has been submitted. You will receive your grade after all steps are complete and your response is fully assessed.

Your response

```
import sys


def decode_token(key: list[str], token: str, is_upper: bool) -> str:
    """
    Decodes a token of message
    Tokens can be generated by tokenize_encoded_message
    Args:
    key (list) : Key used for decoding
    token (str): token to decode
    Returns:
    (str) Decoded token
    """
    key_index = int(token[0]) - 1
    char_index = len(token) - 1
    try:
        result = key[key_index][char_index]
    except IndexError:
        return None
    if is_upper:
        result = result.upper()
    return result
```

```python
def decode_message(key: list, message: str) -> str:
    """
    Decoded a complete message
    Args:
    key (list) : Key used for decoding
    message (str): message to decode
    Returns:
    (str) Decoded Message
    """
    result = ""
    tokenized_msg, is_upper = tokenize_encoded_message(message)
    for token, upper in zip(tokenized_msg, is_upper):
        decoded_token = decode_token(key, token, upper)
        if decoded_token is None:
            return "Error"
        result += decoded_token
    return result


def encode_char(key: list[str], char: str) -> str:
    """
    Encoded one character of message
    Args:
    key (list) : Key used for encoding
    char (str): character to encode
    Returns:
    (str) Encoded word
    """
    result = ""
    if char.isupper():
        result += "0"
    char = char.lower()
    key_index = 0
    char_index = 0
    for i, word in enumerate(key):
        if char in word:
            key_index = i + 1
            char_index = word.index(char) + 1
    if key_index == 0:
        return None
    else:
        result += str(key_index) * char_index + "0"
        return result


def encode_message(key: list[str], message: str) -> str:
    """
    Encodes a complete message
    Args:
    key (list) : Key used for encoding
    message (str): Message to encode
    Returns:
    (str) Encoded string
    """
    result = ""
    for char in message:
        encoded_char = encode_char(key, char)
        if encoded_char is None:
            return "Error"
        else:
            result += encoded_char
    return result


def tokenize_encoded_message(message: str) -> (list[str], list[bool]):
    """
    Tokenizes the encoded message into parts
    Args:
    message (str) : encoded message

    Returns:
    (tuple) : tuple of tokenized words in encoded message
              and is_upper list that indicates whether word is uppercase
              or not i.e. preceded by 0 (other than the delimiter)
    """
    is_upper = []
    tokenized_message = []
    split_message = message.split("0")
    # ignore the last delim
```

```
    del split_message[-1]
    is_upper_ahead = False
    for tok in split_message:
        if tok == "":
            is_upper.append(True)
            is_upper_ahead = True
            continue
        elif not is_upper_ahead:
            is_upper.append(False)
        tokenized_message.append(tok)
        is_upper_ahead = False
    return tokenized_message, is_upper


if __name__ == "__main__":
    with open(sys.argv[1], "r") as file:
        key = file.readline().strip().split(",")
        # sanitize the read key
        # Input containes double quotes (")
        # Need to remove it to make a list of strings
        for i in range(0, len(key)):
            key[i] = key[i].strip().replace('"', "").lower()
        operation = int(file.readline().strip())
        message = file.readline().strip()
    if operation == 2:
        print(decode_message(key, message))
    elif operation == 1:
        print(encode_message(key, message))
```

Test Case Result Breakdown

Test Cases Result: 3 / 3

Test Input	Your Output	Expected Output
"r", "sw", "x", "c", "nm", "q", "yt", "z", "ab" 2 099090770	Bat	Bat
"r", "sw", "x", "c", "nm", "q", "yt", "z", "ab" 1 bat	99090770	99090770
"r", "sw", "x", "g", "nm", "q", "yt", "z", "ab" 1 Cat	Error	Error

Previous Next