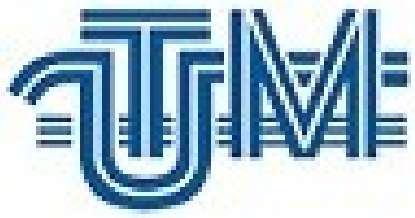

Ministry of Education, Culture and Research of the Republic of Moldova
Technical University of Moldova
Faculty of Computers, Computer Science and Microelectronics
Department of Software Engineering

Report
for laboratory work No. 1
course "Cryptography and Security"

Done by:
Carp Dan-Octavian , gr. FAF-211
Checked by:
Cătălin MÎŢU

Chișinău – 2023

Subject: Caesar's Cipher

Tasks:

1. To implement the Caesar algorithm for the English alphabet in one of the programming languages. Use only the letter encoding as shown in Table 1 (encodings specified in the programming language, eg ASCII or Unicode, are not allowed). Key values will be between 1 and 25 inclusive and no other values are allowed. The text character values are between 'A' and 'Z', 'a' and 'z' and no other values are assumed. If the user enters other values - the correct tuning will be suggested. Before encryption, the text will be converted to uppercase and spaces will be removed. The user will be able to choose the operation - encryption or decryption, enter the key, message or cryptogram and get the decrypted cryptogram or message respectively.
2. To implement the Caesar algorithm with 2 keys, keeping the conditions expressed in Task 1. In addition, key 2 must contain only letters of the Latin alphabet, length no less than 7.

Theory :

In this cipher, each letter of the plaintext is replaced by a new letter obtained by an alphabetical shift. The secret key k, which is the same for encryption as for decryption, consists of the number indicating the alphabetic shift, i.e. $k \in \{1, 2, 3, ..., n-1\}$, where n is the length of the alphabet. The encryption and decryption of the message with the Caesar cipher can be defined by the formulas $c = ek(x) = x + k \pmod n$, $m = dk(y) = y - k \pmod n$, where x and y are the numeric representation of the respective plaintext character. The function called Modulo(a mod b) returns the remainder of dividing the integer a by the integer b.

Implementation:

```
def remove_whitespace(text):
    return "".join(text.split())

def encrypt(text, key):
    result = ""
    for char in text:
        if 'A' <= char <= 'Z':
            result += chr(((ord(char) - ord('A') + key) % 26) + ord('A'))
        elif 'a' <= char <= 'z':
            result += chr(((ord(char) - ord('a') + key) % 26) + ord('a'))
        else:
            print("The text should be in range A - Z or a - z")
            return None
    return result

def decrypt(text, key):
    return encrypt(text, -key)

def main():
    text = input('Enter your text: ').upper()
    text = remove_whitespace(text)
    key = int(input('Enter the key (0 - 25): '))

    if key < 0 or key > 25:
        print('The key should be in range 0 - 25')
        return

    operation = input('Choose "e" for encryption or "d" for decryption: ')

    if operation == 'e':
        result = encrypt(text, key)
        if result:
            print(result)
    elif operation == 'd':
        result = decrypt(text, key)
        if result:
            print(result)
    else:
        print('Invalid operation. Please choose "e" or "d".')

if __name__ == "__main__":
    main()
```

It defines a function remove_whitespace to remove spaces and clean input text.

The encrypt function takes a text and a key as inputs and performs text encryption using the Caesar cipher method.

The decrypt function leverages the encrypt function with a negative key to perform decryption.

The main function orchestrates the program, taking user input for text, encryption/decryption key, and operation (encryption or decryption).

It ensures input validity, handles errors for invalid keys and operations, and prints the result accordingly.

The program is executed when the script is run directly.

Task 2

The code defines two functions, remove_whitespace and generate_new_alphabet, to clean text and create a modified alphabet, respectively.

remove_whitespace removes spaces and joins words in a text input.

generate_new_alphabet processes a second key (key2) to create a new alphabet based on the provided characters, ensuring uniqueness.

The code defines the cipher2 function to perform encryption or decryption using the modified alphabet.

It validates the input parameters, such as key1 (for shift), key2 (for the modified alphabet), and string (for encryption/decryption operation).

The program prints the modified alphabet and then processes the input text character by character, shifting alphabetic characters according to the specified operation and key.

Non-alphabetic characters (e.g., spaces, punctuation) are preserved in the output.

The final result (encrypted or decrypted text) is printed.

A separate function, print_new_alphabet, is used to display the modified alphabet.

User inputs include the text to be processed, the shift key (key1), the characters for the modified alphabet (key2), and the operation (encryption or decryption).

The cipher2 function is called with the provided inputs to perform the encryption or decryption operation based on the modified alphabet.

```python
def remove_whitespace(text):
    return "".join(text.split())

def generate_new_alphabet(key2):
    if not key2.isalpha() or len(key2) < 7:
        return None

    seen = set()
    new_alphabet = ""

    for char in key2:
        char = char.upper()
        if char.isalpha() and char not in seen:
            new_alphabet += char
            seen.add(char)

    for char in "ABCDEFGHIJKLMNOPQRSTUVWXYZ":
        if char not in seen:
            new_alphabet += char

    return new_alphabet

def cipher2(text, key1, key2, string):
    alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
    new_alphabet = generate_new_alphabet(key2)

    if key1 < 0 or key1 > 25:
        print("The first key should be in range 0 - 25")
        return

    if not new_alphabet or len(new_alphabet) != 26:
        print("Key 2 should be a valid permutation of the alphabet with exactly 26 unique characters.")
        return

    print_new_alphabet(new_alphabet)
    result = ""

    for char in text:
        if char.isalpha():
            char = char.upper()
            if string == 'e':
                shifted_index = (new_alphabet.index(char) + key1) % 26
            elif string == 'd':
                shifted_index = (new_alphabet.index(char) - key1) % 26

            shifted_char = new_alphabet[shifted_index]
            if char.islower():
                shifted_char = shifted_char.lower()
            result += shifted_char
        else:
            result += char
```

```
    print(result)

def print_new_alphabet(new_alphabet):
    print(f"Advanced alphabet: {new_alphabet}")

text = input('Enter your text: ').upper()
text = remove_whitespace(text)
key1 = int(input('Enter the first key (0 - 25): '))
key2 = input('Enter key 2 (a string of letters with at least 7 characters): ').upper()
string = input('Choose "e" for encryption or "d" for decryption: ')

cipher2(text, key1, key2, string)
```

Conclusion:

In conclusion, this Caesar cipher implementation allows users to perform text encryption and decryption using a modified alphabet based on two keys: key1 for the shift value and key2 for customizing the alphabet. The code provides clear error messages for invalid inputs, such as keys outside the valid range, insufficient key length, and non-alphabet characters in key2. It guides users through the process of selecting an operation (encrypt or decrypt), inputting keys and messages, and provides the resulting output. The code demonstrates a simple yet effective way to apply the Caesar cipher technique for secure text communication with customization options.