

Ministerul Educatiei, Culturii și Cercetarii al Republicii Moldova Universitatea Tehnic**ă** a Moldovei

Facultatea Calculatoare, Informatic**ă ş**i Microelectronic**ă** Departamentul Ingineria Software și Automatica

Report

for laboratory work No. 1

course "Cryptography and Security"

Done by: Felicia LUPAȘCU, gr. FAF-212

Checked by:  
 Cǎtǎlin 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 :**

The Caesar cipher is a simple encryption technique that was used by Julius Caesar to send secret messages to his allies. It works by shifting the letters in the plaintext message by a certain number of positions, known as the "shift" or "key".

The Caesar Cipher technique is one of the earliest and simplest methods of encryption technique. It's simply a type of substitution cipher, i.e., each letter of a given text is replaced by a letter with a fixed number of positions down the alphabet. For example with a shift of 1, A would be replaced by B, B would become C, and so on. The method is apparently named after Julius Caesar, who apparently used it to communicate with his officials.

Thus to cipher a given text we need an integer value, known as a shift which indicates the number of positions each letter of the text has been moved down.

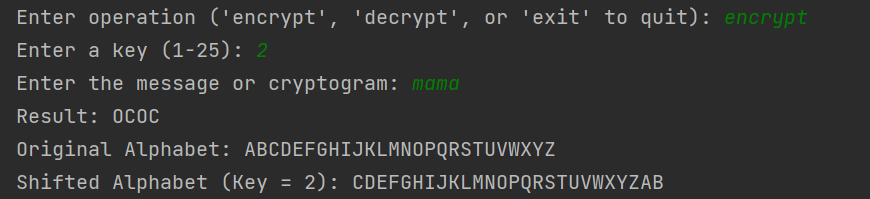
The encryption can be represented using modular arithmetic by first transforming the letters into numbers, according to the scheme, A = 0, B = 1,…, Z = 25. Encryption of a letter by a shift n can be described mathematically as.

**Implementation:**

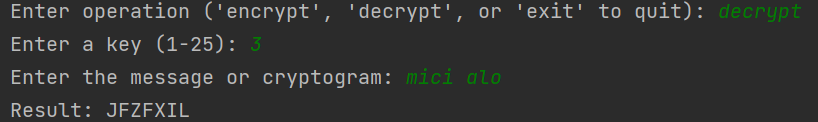
def caesar\_cipher(text, key, operation):  
 def encrypt\_char(char, base, key):  
 return chr((ord(char) - base + key) % 26 + base)  
 def decrypt\_char(char, base, key):  
 return chr((ord(char) - base - key) % 26 + base)  
 result = []  
 for char in text:  
 if char.isalpha():  
 base = ord('A') if char.isupper() else ord('a')  
 if operation == 'encrypt':  
 new\_char = encrypt\_char(char, base, key)  
 elif operation == 'decrypt':  
 new\_char = decrypt\_char(char, base, key)  
 else:  
 return "Invalid operation. Choose 'encrypt' or 'decrypt'."  
 result.append(new\_char)  
 else:  
 return "Invalid input. Only letters are permitted."  
 return ''.join(result)  
def print\_shifted\_alphabet(key, operation):  
 if operation == 'encrypt':  
 alphabet = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'  
 shifted\_alphabet = alphabet[key:] + alphabet[:key]  
 print(f"Original Alphabet: {alphabet}")  
 print(f"Shifted Alphabet (Key = {key}): {shifted\_alphabet}")  
def main():  
 operations = ['encrypt', 'decrypt']  
 while True:  
 operation = input("Enter operation ('encrypt', 'decrypt', or 'exit' to quit): ").lower()  
 if operation == 'exit':  
 print("Done!")  
 break # Exit the loop and end the program  
 if operation not in operations:  
 print("Invalid operation. Only 'encrypt', 'decrypt', or 'exit'.")  
 continue # Continue to the next iteration  
 key\_input = input("Enter a key (1-25): ")  
 if not key\_input.isdigit():  
 print("Invalid key. Please enter a numeric key.")  
 continue # Continue to the next iteration  
 key = int(key\_input)  
 if not (1 <= key <= 25):  
 print("Invalid key. Please select a key within the range of 1 to 25.")  
 continue # Continue to the next iteration  
 text = input("Enter the message or cryptogram: ").replace(" ", "").upper()  
 result = caesar\_cipher(text, key, operation)  
 print(f"Result: {result}")  
 print\_shifted\_alphabet(key, operation)  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

This Python code implements a Caesar cipher, a simple encryption technique. It allows the user to either encrypt or decrypt a message using a specified key, which is an integer between 1 and 25. Here's a breakdown of the code:

* caesar\_cipher: This function takes a text, a key, and an operation ('encrypt' or 'decrypt') as input. It performs the Caesar cipher operation on the text based on the provided key and operation. It returns the encrypted or decrypted result.
* encrypt\_char and decrypt\_char functions: These inner functions are used to encrypt and decrypt individual characters of the text based on the given key and operation.
* print\_shifted\_alphabet: This function prints the original and shifted alphabets, but only for the encryption operation. It calculates the shifted alphabet based on the key.
* main: This is the main program loop. It allows the user to choose between 'encrypt' and 'decrypt' operations, specify a key, and provide a message or cryptogram for encryption or decryption. It also handles input validation to ensure that the user provides valid inputs.

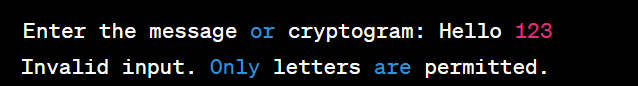
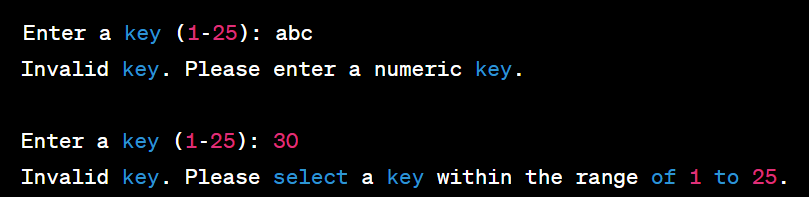
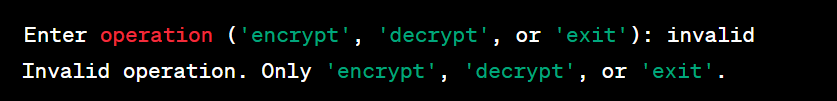


For the "encrypt" operation in the provided Caesar cipher code, here's the expected output and its description: **Input Prompt:** The program first asks the user to enter an operation ('encrypt', 'decrypt', or 'exit'). **Operation Prompt:** If the user chooses 'encrypt' by typing "encrypt" (case-insensitive), the program proceeds to the next step. **Key Input Prompt:** The user is then prompted to enter a key (an integer between 1 and 25, inclusive).**Text Input Prompt**: After entering the key, the user is prompted to enter the message or cryptogram they want to encrypt. Spaces are removed, and the input is converted to uppercase**. Result:** The program performs the Caesar cipher encryption using the provided key and the input text. It then prints the encrypted result. **Shifted Alphabet:** Additionally, if it's an encryption operation, the program prints both the original alphabet and the shifted alphabet based on the provided key. This part of the output is only shown for encryption.



The user chooses the "decrypt" operation during program execution.After selecting "decrypt," the program prompts the user to enter a key (an integer between 1 and 25).The user is then asked to input the encrypted message or cryptogram they want to decrypt.The program performs the Caesar cipher decryption using the provided key and the input text. It prints the decrypted result.

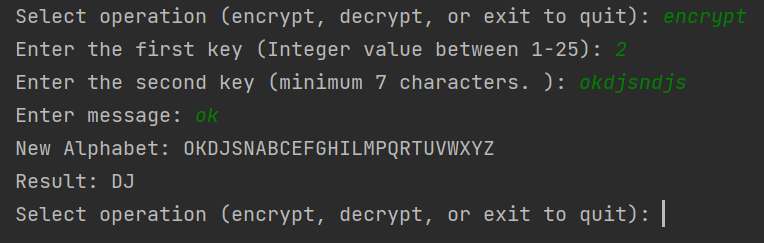
Here are **examples of the errors generated when incorrect input** is provided in the provided Caesar cipher code:

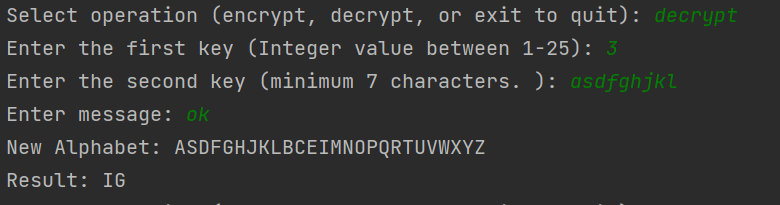


**Task 2:**

def generate\_new\_alphabet(key2):  
 key2 = ''.join(sorted(set(key2.upper()), key=key2.upper().index))  
 alphabet = ''.join(chr(ord('A') + i) for i in range(26))  
 new\_alphabet = key2 + ''.join(char for char in alphabet if char not in key2)  
 print(f'New Alphabet: {new\_alphabet}')  
 return new\_alphabet  
def caesar\_cipher(input\_text, key1, key2, operation):  
 new\_alphabet = generate\_new\_alphabet(key2)  
 if not (1 <= key1 <= 25):  
 return 'Invalid key. Key must be between 1 and 25.'  
 modified\_text = ''.join(input\_text.split()).upper()  
 result = ''  
 for char in modified\_text:  
 if char in new\_alphabet:  
 index = new\_alphabet.index(char)  
 if operation == 'encrypt':  
 new\_index = (index + key1) % len(new\_alphabet)  
 elif operation == 'decrypt':  
 new\_index = (index - key1 + len(new\_alphabet)) % len(new\_alphabet)  
 else:  
 return 'Invalid operation.Operation must be encrypt, decrypt, or exit.'  
 result += new\_alphabet[new\_index]  
 else:  
 return 'Invalid character. Only alphabet characters (A-Z) are allowed.'  
 return result  
while True:  
 operation = input('Select(encrypt, decrypt, or exit to quit): ').strip().lower()  
 if operation == 'exit':  
 print('Exiting the program. Goodbye!')  
 break  
 elif operation not in ('encrypt', 'decrypt'):  
 print('Invalid operation. Please enter encrypt, decrypt, or exit to quit.')  
 continue  
 key1\_input = input('Enter the first key (Integer value between 1-25): ').strip()  
 try:  
 key1 = int(key1\_input)  
 if not (1 <= key1 <= 25):  
 print('Invalid key. Key must be between 1 and 25.')  
 continue  
 except ValueError:  
 print('Invalid key. Key must be an integer between 1 and 25.')  
 continue  
 key2 = input('Enter the second key (minimum 7 characters. ): ').strip()  
 if len(key2) < 7:  
 print('Invalid key. Second key must be at least 7 characters long.')  
 continue  
 message = input('Enter message: ')  
 result = caesar\_cipher(message, key1, key2, operation)  
 print(f'Result: {result}')

This code is a Caesar cipher program that allows encryption and decryption of messages with customizable keys. The output of this code is a Caesar cipher encrypted or decrypted message, depending on the user's choice, using specified keys and a modified alphabet. It also provides feedback messages for input validation and guidance throughout the process.





Examples of errors when introducing wrong input for key2 (the second key) could include:

**Entering fewer than 7 characters for key2**, which should result in the error message: "Invalid key. Second key must be at least 7 characters long."**Including non-alphabet characters** in key2, which should trigger the error message: "Invalid character. Only alphabet characters (A-Z) are allowed."**Having duplicate characters** in key2, which should still be removed but without any specific error message, as duplicates are silently removed during the processing of key2.

**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.