

Ministry of Education and Culture of the Republic of Moldova

Technical University of Moldova

Department of Software and Automation Engineering

**REPORT**

Laboratory work No. 1

**Discipline**: Cryptography and Security

Elaborated: Șarov Andrei FAF-222,

Checked: Nirca Dumitru asist. univ.,

Chișinău 2024

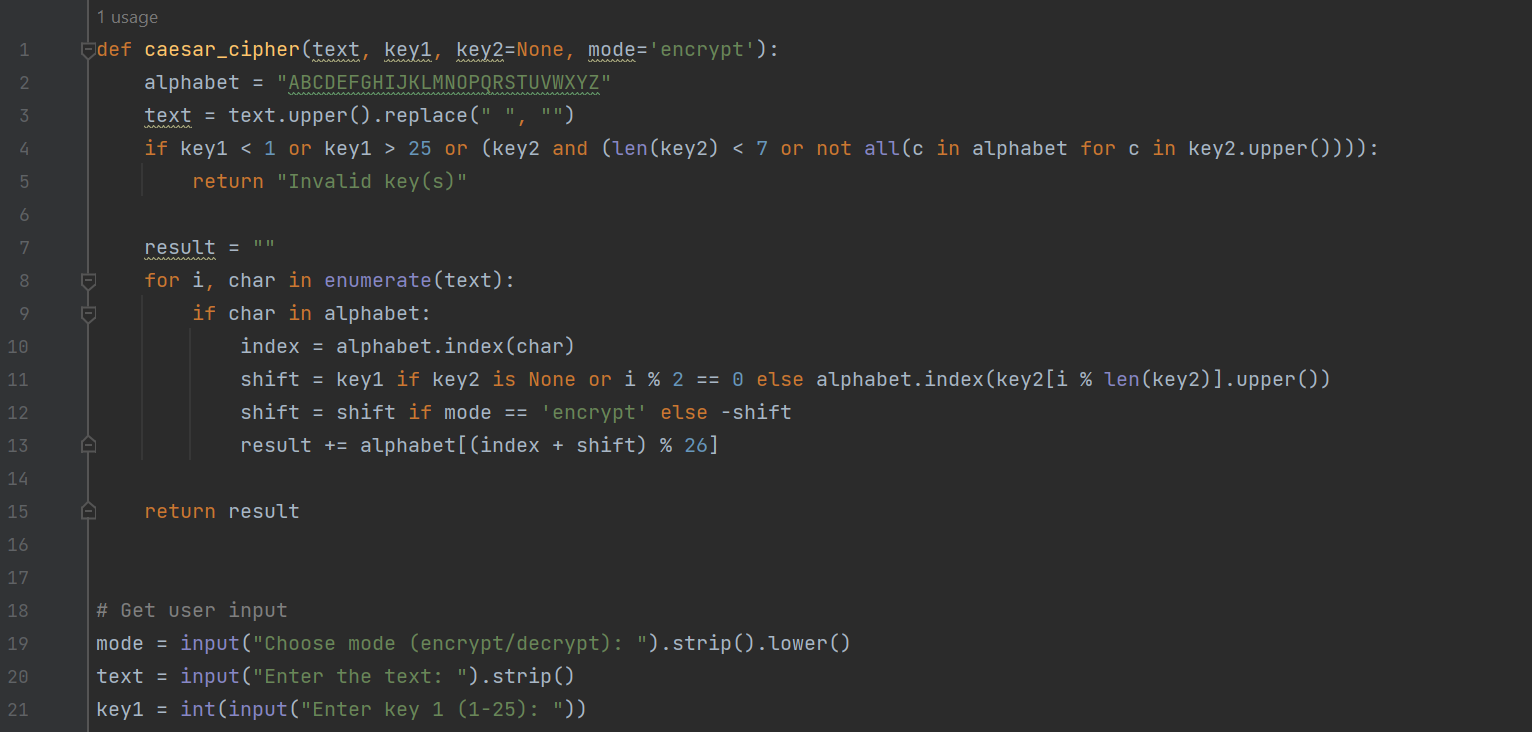
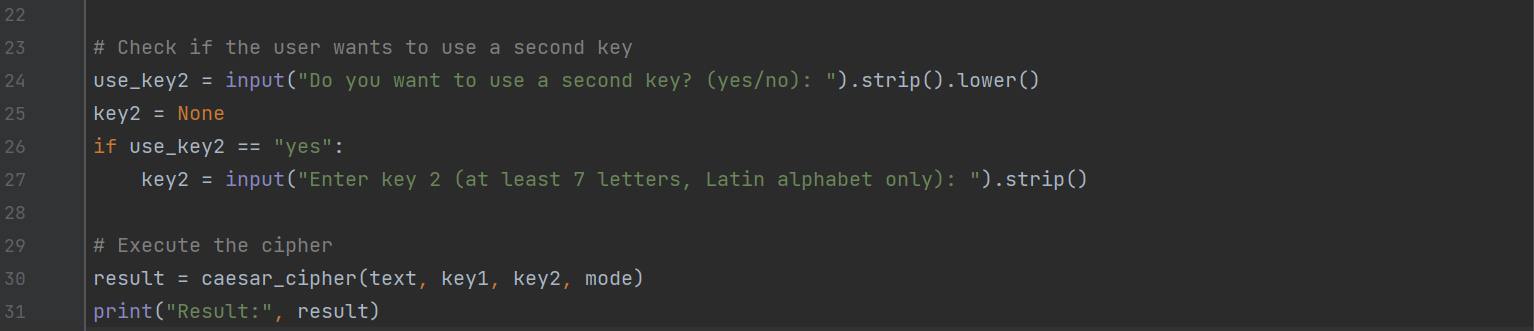
## Topic: Caesar’s Cipher

## Tasks:

1. Implement the Caesar algorithm for the English alphabet in one of the programming languages. Use only the letter encodings as shown in Table 1 (encodings specified in the programming language, e.g. ASCII or Unicode, are not allowed to be used). Key values shall be between 1 and 25 inclusive and no other values are allowed. Text character values shall be between 'A' and 'Z', 'a' and 'z' and no other values are prefixed. If the user enters other values - the user will be prompted for the correct slide. Before encryption the text will be converted to upper case 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 cryptogram or decrypted message respectively.
2. Implement the Caesar algorithm with 2 keys, preserving the conditions expressed in Task 1.1. In addition, key 2 must contain only letters of the Latin alphabet, and have a length of not less than 7

## Theoretical notes:

The Caesar cipher is one of the oldest encryption techniques, shifting letters by a fixed number (key) to create a new text (ciphertext). Due to its simplicity, the Caesar cipher offers only basic security and is prone to brute-force attacks, as only 25 possible shifts exist. In cryptographic terms, Caesar is a substitution cipher because it replaces each letter with another. In the extended form (Task 2), Caesar uses two keys, offering more complexity and potentially higher security by alternating shifts based on an additional key.

**Implementation:******

1. The code defines the Caesar cipher function, caesar\_cipher, to handle both single-key and two-key encryption and decryption. The function takes the input text, key1, an optional key2, and a mode indicating encryption or decryption. The alphabet is predefined as a string of uppercase letters, allowing letter indexing without ASCII or Unicode.
2. The function first processes the input text by converting it to uppercase and removing spaces. Then, it checks that key1 is within the range of 1 to 25. If key2 is provided, it must be at least seven characters and consist only of Latin letters. If any conditions are not met, the function returns "Invalid key(s)".
3. The function then iterates over each character in the text. For each letter found in the alphabet, it determines the position within the alphabet and calculates the shift. For a single key, the shift is determined solely by key1. In the two-key scenario, characters alternate between key1 and shifts based on the corresponding letter in key2. The shift is applied positively for encryption and negatively for decryption. The modulo 26 operation ensures the resulting shift stays within the alphabet's bounds.
4. The shifted character is then added to the result string, which accumulates the final output. Once all characters are processed, the function returns the completed encryption or decryption text.
5. The main code block allows for user input. It prompts the user for the mode, text, and key1. If the user chooses to use a second key, it requests key2. Finally, it calls the caesar\_cipher function with the given inputs and prints the resulting encrypted or decrypted message. This interactive design makes the cipher usable directly from the command line.

**Conclusions:**

This laboratory exercise explored both single and double-key Caesar cipher implementations. Understanding the Caesar cipher introduces fundamental cryptographic concepts like substitution, modular arithmetic, and the idea of keys in encryption. While historically significant, the Caesar cipher’s limitations highlight the need for more sophisticated algorithms in modern cryptography. This exercise reinforces key cryptographic principles and introduces additional complexity through multi-key encryption, making it a useful stepping stone to advanced ciphers.