MINISTRY OF EDUCATION AND RESEARCH OF REPUBLIC OF MOLDOVA
TECHNICAL UNIVERSITY OF MOLDOVA
FACULTY OF COMPUTERS, INFORMATICS AND MICROELECTRONICS
DEPARTMENT OF SOFTWARE AND AUTOMATION ENGINEERING

ANALYSIS OF ALGORITHMS
LABORATORY WORK #5

Cifrul lui Cezar: Implementare și Extensie

Author:	*Verified:*
Andrei Chicu	The Prof
std. gr. FAF–233	Department of SEA, FCIM UTM

Chișinău, 2025

1 Introduction

github url: `https://github.com/...`

1.1 Objective

The objective of this laboratory work is to implement the Caesar Cipher algorithm, covering both the standard fixed-shift version and an extended version that uses a keyword to permute the alphabet, significantly increasing the cipher's key space and cryptoresistance.

1.2 Tasks

- Task 1.1: Standard Caesar Cipher (Single Key k_1)
- Task 1.2: Permutation Caesar Cipher (Two Keys k_1 and k$_2$)
- Task 1.3: Cipher Verification (Exchange and Decrypt)

1.3 Theoretical Notes

The standard Caesar cipher uses the formulas:

- Encryption: $ck(x) = (x + k)(modn)$
- Decryption: $mk(y) = (yk)(modn)$ where $n = 26$ for the English alphabet and the shift key $k \in 1, 2, \ldots, 25$.
- The permutation cipher modifies this by using a new alphabet sequence defined
  by the keyword k_2.

1.4 Task 1.1: Standard Caesar Cipher (Single Key)

1.4.1 Implementation Details

The standard Caesar Cipher implementation uses a single integer key, k_1, for the shift.

1. Key and Text Validation

   `getShiftKey`: Validates that the shift key k_1 is an integer in the range $[1, 25]$.

   `sanitizeText`: Ensures the input plaintext is converted to uppercase and all non-letter characters (including spaces) are removed.

2. Cipher Logic The core logic resides in the `processText` function, which uses the constant `alphabet` The encryption/decryption is achieved by calculating the index of the new character using modular arithmetic:

   - Find the index of the character x in the standard alphabet.
   - For encryption, calculate `(index+k) (mod 26)`.
   - For decryption, calculate `(index-k) (mod 26)`. The result is adjusted to ensure it remains positive (e.g., `(index-k+26) (mod 26)`).

1.5 Task 1.2: Permutation Caesar Cipher (Two Keys)

1.5.1 Implementation Details

This extended cipher uses a shift key k_1 (integer) and a permutation key k_2 (keyword string).

1. Key Validation `getPermutationKey`: Validates the permutation keyword k_2:

   - Must be composed only of letters.
   - Must have a minimum length of 7 characters.

2. Permuted Alphabet Generation The `generatePermutedAlphabet` function is responsible for creating the new alphabet based on k_2: The keyword k_2 is sanitized and uppercased. Unique letters from k_2 are appended to the new alphabet in the order they first appear (duplicates are excluded). The remaining letters of the standard alphabet (A-Z) that were not in the keyword are appended in their natural order.

   Example: For $k_2 = "cryptography"$, the permuted alphabet is `CRYPTOGAHBDEFIJKMLNQSUVWXZ`.

3. Cipher Logic The same `processText` function is reused, but it now operates on the permuted alphabet string instead of the standard one. The indices are mapped based on the position within this new 26-character sequence.

1.6 Task 1.3: Cipher Verification

1.6.1 Implementation Details

This task verifies the practical application of the Permutation Caesar Cipher through a peer exchange.

1. Exchange Results

1.7 Conclusions

The laboratory work successfully implemented the Caesar Cipher and its extension. The use of a permutation keyword significantly complicates an exhaustive key search compared to the standard version, although the cipher remains vulnerable to frequency analysis. All requirements regarding text sanitization (uppercase, no non-letters) and key validation ($k_1 \in [1, 25], len(k_2) \geq 7$) were met in the Go implementation.