Title:

Development of *Caesar Cipher program* for encrypted text encoding and decoding

Team members:

Laiba (CT-060)

Areeba (CT-070)

Minahil (CT-066)

INTRODUCTION:

The Caesar Cipher is one of the oldest and simplest encryption techniques, used by Julius Caesar to protect military messages. This project involves creating a console-based application that can encrypt and decrypt messages using this historical algorithm, providing a hands-on introduction to the fundamentals of cryptography and encryption.

OBJECTIVE:

- Change normal text into secret code using the Caesar Cipher method
- Bring it back to the original text
- Help in understanding the basics of cryptography and how old military encryption methods worked.
- Easy to use, works with user input
- Shows the difference between encryption and decryption.

TECHNICAL APPROACH:

1. Programming Language:

The tool will be implemented in C because of its efficiency, simplicity, and ability to handle string manipulation through character arrays.

2. Core Functions:

void encrypt (char message[], int key)

- Loops through the string and shifts each alphabetic character forward by the key.

void decrypt (char message[], int key)

- Shifts characters backward by the key to recover plain text.

void bruteForce(char message[])

- Tries all 25 possible shifts and prints each possible decrypted result.

3. Algorithm Design

Use ASCII values for characters:

- Uppercase letters: A–Z (65–90)
- Lowercase letters: a–z (97–122)

Apply Caesar cipher formula:

- *Encryption:* (ch - base + key) % 26 + base
- *Decryption:* (ch - base - key + 26) % 26 + base
- base is 65 for uppercase and 97 for lowercase.

4. ***Testing and Validation:***

Test with:

- Mixed-case text (Hello World)
- Large keys (key = 52, which should behave like key = 0)
- Non-alphabetic characters (Hello123!)

Validate that encryption and decryption are exact opposites.

5. ***Scalability***

Future extensions can include:

- Reading/writing encrypted text to files.
- Adding support for other classical ciphers (Vigenère, Playfair).

IMPLEMENTATION PLAN:

Week 1: Basic program structure and user input handling.

Week 2: Implementation of the encryption function.

Week 3: Implementation of the decryption function and menu system.

Week 4: Testing, debugging, and final improvements.

EXPECTED CHALLENGES:

- Handling wrap-around for letters at the end of the alphabet (e.g., 'z' shifted by 1 becomes 'a').
- Preserving non-alphabetic characters like spaces and punctuation.
- Input validation for the shift key.

EXPECTED OUTCOMES:

Upon completion, the program will successfully:

- Encrypt normal texts and messages into a secret code
- Decrypt secret code back to normal text
- Handles edge cases (letter wrap around z-a)
- Works with any shift key value
- Display clear results on the console

CONCLUSION:

This project will improve our coding skills while demonstrating how to implement classical cryptographic techniques using basic programming concepts.