Mini-project set $n°1$: **Functions & Procedures**

1 Caesar Cipher Encryption and Decryption

Objective

Write a C program that encrypts and decrypts messages using the **Caesar Cipher** algorithm[1].

Background

The Caesar Cipher is a substitution cipher that shifts letters by a fixed number of positions in the alphabet. For example, with a shift of 3:

- 'A' → 'D', 'B' → 'E', ..., 'Z' → 'C'

Decryption reverses the shift. The program should be cases-sensitive and ignore non-alphabetic characters.

Tasks

1. **Implement Encryption Function** Write a function `cesarEncrypt` that takes a string and an integer `key` (shift value) and returns the encrypted message.

   Input: "HELLO", key = 3
   Output: "KHOOR"

2. **Implement Decryption Function** Write a function `cesarDecrypt` that reverses the encryption process using the same `key`.

   Input: "KHOOR", key = 3
   Output: "HELLO"

3. **Main Program Implementation** Implement a `main` function that:
   - Prompts the user for a message and a shift value,
   - Encrypts and displays the result,
   - Decrypts the result and verifies it matches the original message.

4. **Testing and Validation** Test your program with:
   - Uppercase/lowercase letters (e.g., "Hello"),
   - Non-alphabetic characters (e.g., spaces, numbers),
   - Large shift values (e.g., key = 27).

[1] *Bonus points will be awarded for implementing additional features.*

2 Wordle-Inspired Guessing Game

Objective

Write a C program that implements a game inspired by **Wordle**. The player must guess a secret word within a limited number of attempts. The program provides feedback on correct letters, misplaced letters, and incorrect letters[1].

Game Rules

1. The program selects a secret word (e.g., `"CODE"`).
2. The player has a limited number of attempts (e.g., 6).
3. After each guess, the program displays:
   - ✓ for correct letters in the right position,
   - ? for correct letters in the wrong position,
   - × for letters not in the word.
4. The player wins if they guess the word before running out of attempts.

Tasks

1. **Implement the `checkWord` Function** Write a function `checkWord` that compares the player's guess with the secret word and returns feedback symbols (✓, ?, ×).

   Secret word: "CODE"
   Player's guess: "COTE"
   Feedback: ✓ × ✓ ✓

2. **Design the Game Loop**
   - Prompt the player to enter a guess.
   - Limit the number of attempts (e.g., 6).
   - Display feedback after each guess.
   - End the game if the word is guessed or attempts are exhausted.
3. **Test with Multiple Words**
   - Store a list of secret words and select one randomly.
   - Allow the player to restart the game after winning/losing.

[1] *Bonus points will be awarded for implementing additional features.*