**Technical Skills Survey Project Part 2- Helping Caesar's Traditional Secret Party**

# **Objective 🎯**

Your task is to craft two intricate functions: one that encrypts a message in the spirit of Caesar's Cipher but with added layers of complexity and another that decrypts such messages, unveiling their hidden content.

# **Starter Code 🌱**

const alphabet = "abcdefghijklmnopqrstuvwxyz";

function encrypt (message, shiftValue)

{

// Your encryption code here

return encryptedMessage;

}

function decrypt (encryptedMessage, shiftValue)

{

// Your decryption code here

return decryptedMessage;

}

# **Requirements 📋**

1. **Implementing the Encryption Algorithm of Caesar's Cipher**
   * Step 1: Take a plaintext message and a shift value and return an encrypted string. If the message includes a character out of the alphabet, pass it as is to the encrypted string.
   * Step 2: After every two letters, insert a random letter from the alphabet.
2. **Implementing the Decryption Algorithms of Caesar's Cipher**
   * Take in the encrypted message and a shift value and return the original plaintext message.
   * Accurately reverse the encryption process to retrieve the original message.
3. **Decrypting the Secret Message**
   * Iueuan jrxuq cjythdykwxaj mixkqtaeml ebv wHenckvbkei rqdmt fHukckvi.r Jbxuihus, tmxayiwfuxh sjxau amenhtv 'zQkhhuubyjkit' yjew jhxux mxydatij. zJxmu hvymhihj ajel kldlsuyjb dyju yid uekdh qIbkqsxa xsxqqdvduzb wuqzhdoi qjxwu waueo xjem jfxuy dpuntj dgkvuiwj.
   * Decrypt the above secret message using 42 as the shift value and complete the quest.
4. **Using Comments**: As you build your functions, document your thought process, the purpose of each section of your code, and any challenges you come across. Good commenting not only helps others understand your code but also aids you in tracking your logic.
5. **Attributing Help**:
   * If you incorporate code or inspiration from online resources, attribute the source. At a minimum, provide the URL.
   * If you get help from mentors or TAs, describe the help given and attribute the name.

# **Expected Output 🖼️**

A set of functions that can seamlessly encrypt and decrypt messages with the given complexities.

# **Hints 💡**

## **General Hints:**

* Break the problem down and tackle one aspect of the task at a time.
* Use helper functions to make your code more modular and readable.
* Test your functions with various input strings to ensure accuracy.

## **Step 1 Hints:**

* **Shift Only**: At this point, focus only on shifting the characters based on the shift value. You don't need to worry about random characters or alternating shift directions yet.
* Remember to handle both uppercase and lowercase characters.
* If a character isn't in the alphabet, just add it to the encrypted/decrypted message as is.
* Remember to handle negative indices after shifting. The modulo operation can return a negative result in some languages, including JavaScript.
* **Example**: For a shift value of 42, if your encrypted message is "Xubbe Rhkjki, cuuj cu qj jxu xywx wqhtudi.", decrypting it should yield "Hello Brutus, meet me at the high gardens.".

## **Step 2 Hints:**

* **Introduce Random Letters**: After every two characters in the encrypted message, insert a random letter. This will make your encrypted message longer than the original.
* When decrypting, you'll need a strategy to skip over these random letters to get back the original message.
* Keep a counter to track every two characters. Reset this counter after inserting a random letter.
* Be careful when skipping over random letters during decryption. Ensure you don't accidentally skip over a valid encrypted character.
* **Example**: For a shift value of 42, if your encrypted message is "Xuobbce eRhakjikiw, gcueujr cfu wqjy jzxul xfywox pwqghtiudri.", decrypting it should yield "Hello Brutus, meet me at the high gardens.".

Answer:

const alphabet = "abcdefghijklmnopqrstuvwxyz";

// Helper function to shift a character by shiftValue

function shiftCharacter(char, shiftValue) {

const isUpperCase = char === char.toUpperCase();

const normalizedChar = char.toLowerCase();

// Find the index of the character in the alphabet

const index = alphabet.indexOf(normalizedChar);

// If character is not in the alphabet, return it as is

if (index === -1) return char;

// Compute new index with shiftValue (use modulo to wrap around)

const newIndex = (index + shiftValue) % alphabet.length;

const shiftedChar = alphabet[newIndex < 0 ? newIndex + alphabet.length : newIndex];

// Preserve original case

return isUpperCase ? shiftedChar.toUpperCase() : shiftedChar;

}

function encrypt(message, shiftValue) {

let encryptedMessage = "";

let count = 0;

for (let i = 0; i < message.length; i++) {

const shiftedChar = shiftCharacter(message[i], shiftValue);

encryptedMessage += shiftedChar;

count++;

// After every two characters, add a random character

if (count % 2 === 0 && i < message.length - 1) {

const randomChar = alphabet[Math.floor(Math.random() \* alphabet.length)];

encryptedMessage += randomChar;

}

}

return encryptedMessage;

}

function decrypt(encryptedMessage, shiftValue) {

let decryptedMessage = "";

let count = 0;

for (let i = 0; i < encryptedMessage.length; i++) {

count++;

// Skip every third character (the random one)

if (count % 3 === 0) {

continue;

}

// Shift character back to decode

decryptedMessage += shiftCharacter(encryptedMessage[i], -shiftValue);

}

return decryptedMessage;

}

const encrypted = encrypt("Hello Brutus, meet me at the high gardens.", 42);

console.log("Encrypted Message:", encrypted);

// Decrypt the provided secret message

const decrypted = decrypt(

"Iueuan jrxuq cjythdykwxaj mixkqtaeml ebv wHenckvbkei rqdmt fHukckvi.r Jbxuihus, tmxayiwfuxh sjxau amenhtv 'zQkhhuubyjkit' yjew jhxux mxydatij. zJxmu hvymhihj ajel kldlsuyjb dyju yid uekdh qIbkqsxa xsxqqdvduzb wuqzhdoi qjxwu waueo xjem jfxuy dpuntj dgkvuiwj.",

42

);

console.log("Decrypted Message:", decrypted);