WEEK– 02

Task – 1

Aim: To perform encryption and decryption using the Caesar cipher substitution technique

Description: one of the simplest and oldest methods of encrypting messages named after Julius Caesar. This technique involves shifting the letters of the alphabet by a fixed number of places.

Formula:

Encryption:
E(x) = (x + k) mod 26

Decryption:
D(x) = (x - k) mod 26
Where:

x is the position of the character (0 to 25 for A to Z)

k is the key (number of positions to shift

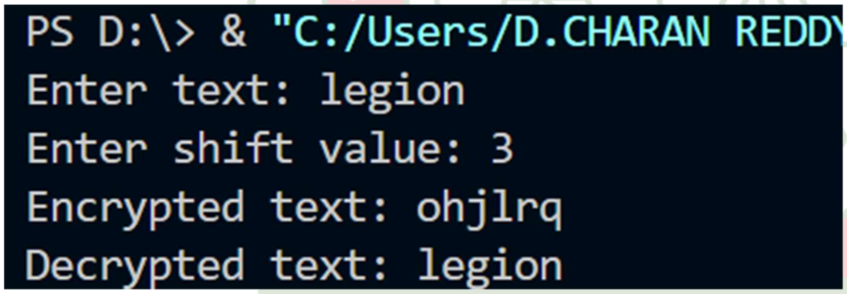
Algorithm:

```
1) Define the alphabet.
2) For each character in the message:
        • If it is a letter, find its index in the alphabet.
        • Apply the shift (+k for encryption, -k for decryption).
        • Use modulo to wrap around the alphabet.
        • Replace it with the new letter.
3) Leave non-alphabet characters unchanged (optional).
```

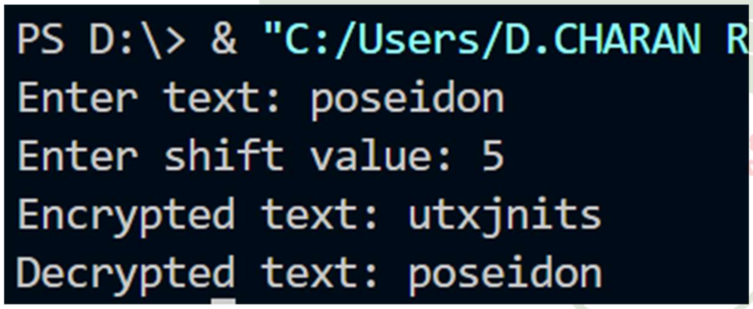
Program:

```
def caesar(text, shift):
    result = ""
    for char in text:
        if char.isalpha():
            shift_base = ord('A') if char.isupper() else ord('a')
            encrypted_char = chr((ord(char) - shift_base + shift) % 26 + shift_base)
            result += encrypted_char
        else:
            result += char
    return result

if __name__ == "__main__":
    text = input("Enter text: ")
    shift = int(input("Enter shift value: "))
    encr = caesar(text, shift)
    decr = caesar(encr, -shift)
    print("Encrypted text:", encr)
    print("Decrypted text:", decr)
```

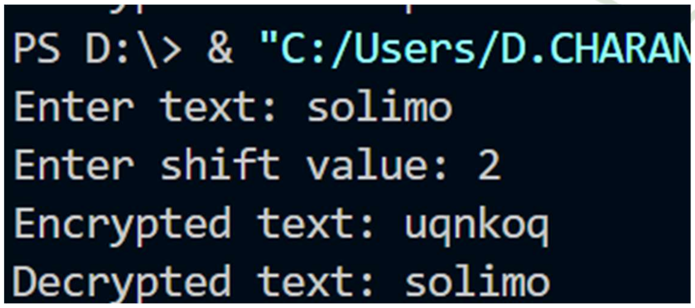
Output:



```
PS D:\> & "C:/Users/D.CHARAN REDDY
Enter text: legion
Enter shift value: 3
Encrypted text: ohjlrq
Decrypted text: legion
```



```
PS D:\> & "C:/Users/D.CHARAN R
Enter text: poseidon
Enter shift value: 5
Encrypted text: utxjnits
Decrypted text: poseidon
```



```
PS D:\> & "C:/Users/D.CHARAN
Enter text: solimo
Enter shift value: 2
Encrypted text: uqnkoq
Decrypted text: solimo
```

Output Analysis:

In the given output, all three test cases demonstrate correct encryption and decryption using a Caesar cipher. In Test Case 1, the lowercase text legion is shifted 3 positions forward to become ohjlrq, and correctly decrypted back. Test Case 2 shifts poseidon by 5 to produce utxjnits, with decryption restoring the original text. In Test Case 3, solimo is shifted by 2 to become uqnkoq, and decryption works accurately. All characters are lowercase, no spaces or special characters are involved, and the cipher consistently applies the shift logic, confirming correct handling of alphabetic characters.

Conclusion:

The Caesar cipher code accurately performs encryption and decryption by shifting alphabetic characters while preserving the case. It handles both uppercase and lowercase letters correctly and leaves non-alphabetic characters like spaces, digits, and punctuation unchanged. The results from all three test cases confirm the code's correctness and reversibility.

TASK-02

Aim: To perform encryption and decryption using the Playfair cipher substitution technique.

Description: Playfair cipher was the first practical diagraph substitution cipher. The scheme was invented in 1854 by Charles Wheatstone but was named after lord Playfair who promoted the use of the cipher.

Algorithm:

1) **Generate Key Matrix:**
   - Convert key to uppercase, replace J with I.
   - Remove duplicates, fill remaining letters (A–Z, skipping J) to make a 5×5 matrix.
2) **Prepare Plaintext:**
   - Convert to uppercase, replace J with I.
   - Split into digraphs (pairs). Insert X between repeated letters or at the end if odd-length.
3) **Encrypt Each Digraph:**
   - Same row: Replace each letter with the one to its right (wrap around).
   - Same column: Replace each letter with the one below (wrap around).
   - Different row & column: Replace each letter with the letter in the same row but other's column.
4) **Decrypt:**
   - Use same rules but reverse the direction (left/up instead of right/down).

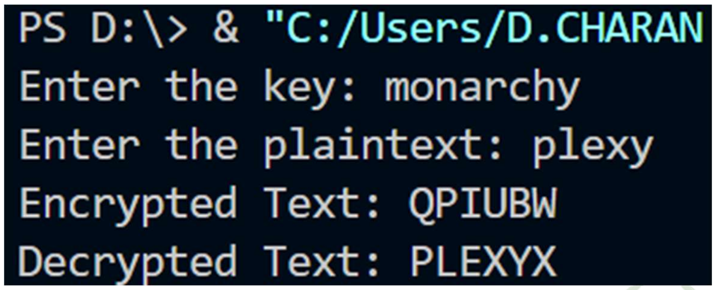
Output the Encrypted or Decrypted Text

Program:

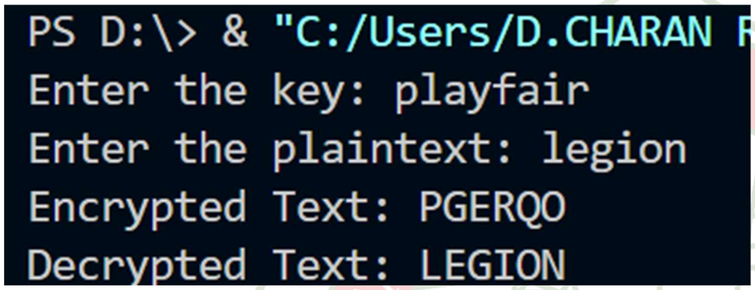
```
def matrix(key):
    key = key.upper().replace('J', 'I')
    seen, box = "", []
    for c in key + "ABCDEFGHIKLMNOPQRSTUVWXYZ":
        if c not in seen and c.isalpha():
            seen += c
    for i in range(0, 25, 5):
        box.append(list(seen[i:i+5]))
    return box
```

```
def locate(c, box):
    for i in range(5):
        if c in box[i]:
            return i, box[i].index(c)
    return None

def prepare(text):
    text = text.upper().replace('J', 'I')
    text = ''.join(c for c in text if c.isalpha())
    res, i = "", 0
    while i < len(text):
        a = text[i]
        b = text[i+1] if i+1 < len(text) else 'X'
        if a == b:
            res += a + 'X'
            i += 1
        else:
            res += a + b
            i += 2
    if len(res) % 2:
        res += 'X'
    return res

def playfair(text, key, mode='encrypt'):
    box = matrix(key)
    if mode == 'encrypt':
        text = prepare(text)
    else:
        text = text.upper().replace('J', 'I')
        text = ''.join(c for c in text if c.isalpha())

    res = ""
    shift = 1 if mode == 'encrypt' else -1
    for i in range(0, len(text), 2):
        a, b = text[i], text[i+1]
        loc1 = locate(a, box)
        loc2 = locate(b, box)
        if not loc1 or not loc2:
            continue
        r1, c1 = loc1
        r2, c2 = loc2
        if r1 == r2:
            res += box[r1][(c1 + shift) % 5] + box[r2][(c2 + shift) % 5]
        elif c1 == c2:
            res += box[(r1 + shift) % 5][c1] + box[(r2 + shift) % 5][c2]
        else:
            res += box[r1][c2] + box[r2][c1]
    return res

key = input("Enter the key: ")
text = input("Enter the plaintext: ")

enc = playfair(text, key, 'encrypt')
print("Encrypted Text:", enc)

dec = playfair(enc, key, 'decrypt')
print("Decrypted Text:", dec)
```

Output :



```
PS D:\> & "C:/Users/D.CHARAN
Enter the key: monarchy
Enter the plaintext: plexy
Encrypted Text: QPIUBW
Decrypted Text: PLEXYX
```



```
PS D:\> & "C:/Users/D.CHARAN F
Enter the key: playfair
Enter the plaintext: legion
Encrypted Text: PGERQO
Decrypted Text: LEGION
```

Output Analysis:

In both test cases, the Playfair cipher is used for encryption and decryption with a keyword-based 5x5 matrix. In Test Case 1, using the key "monarchy", the plaintext "plexy" is converted to "QPIUBW". Since Playfair requires even-length input, a filler character 'X' is added, resulting in the decrypted text "PLEXYX", showing correct padding and decryption.

In Test Case 2, the key "playfair" is used to encrypt "legion" into "PGERQO", and it is accurately decrypted back to "LEGION". The encryption handles letter pair rules (same row, column, or rectangle), and decryption confirms the algorithm functions as intended. All transformations preserve the structure required by the Playfair cipher, and the case of the output is standardized to uppercase, showing consistent formatting.

Conclusion:

The Playfair cipher code provides a reliable implementation of classical digraph-based encryption. It correctly processes the input by converting text to uppercase, replacing 'J' with 'I', removing non-alphabetic characters, and forming digraphs while handling duplicate letters and odd-length inputs by inserting 'X'. The encryption and decryption functions apply standard Playfair rules for same row, same column, and rectangle cases, ensuring accurate and reversible transformations. Overall, the code demonstrates a clear and effective application of the Playfair cipher, producing consistent results across various test cases.