Name: Parth Das
Sap: 60004220185
Roll No: C-111
Subject: Information Security

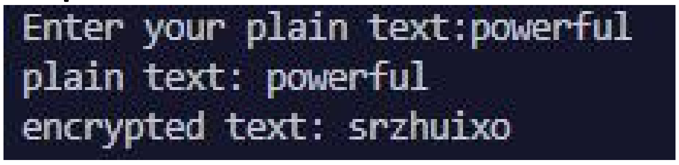
Topic:- To Implement Playfair and Caesar Cipher

1. Caesar Cipher

Code:-

```
message = input('Enter your plain text:')
shift = 3

def caesar(message, shift):
   alphabet = 'abcdefghijklmnopqrstuvwxyz'
   encrypted_text = ''

   for char in message.lower():
      if char == ' ':
         encrypted_text += 'p'
      else:
         index = alphabet.find(char)
         new_index = (index + shift) % len(alphabet)
         encrypted_text += alphabet[new_index]
   print('plain text:', message)
   print('encrypted text:', encrypted_text)

caesar(message, shift)
```

Output:-



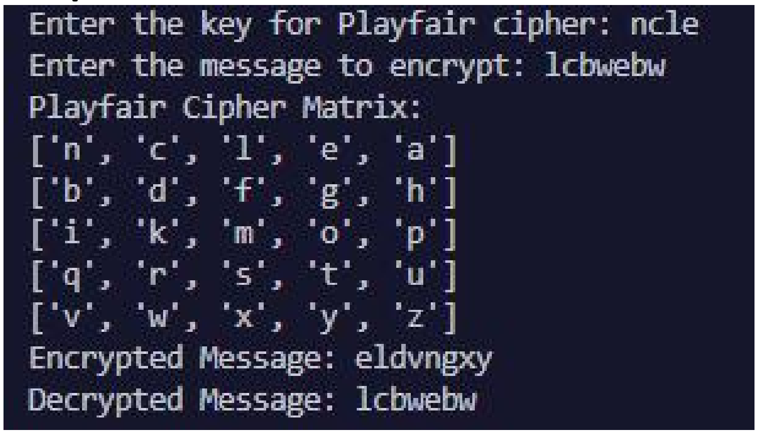
```
Enter your plain text:powerful
plain text: powerful
encrypted text: srzhuixo
```

2. PlayFair Cipher
Code:-

```
def create_playfair_matrix(key):
    key = key.replace('j', 'i').lower()
    matrix = []
    seen = set()

    # Create a 5x5 matrix
    for char in key:
        if char not in seen and char.isalpha():
            matrix.append(char)
            seen.add(char)

    # Add remaining characters of the alphabet to the matrix
    alphabet = 'abcdefghiklmnopqrstuvwxyz'  # j is merged with i
    for char in alphabet:
        if char not in seen:
            matrix.append(char)
            seen.add(char)

    # Return the matrix as a list of lists (5x5 grid)
    return [matrix[i:i + 5] for i in range(0, len(matrix), 5)]

# Function to get the position of a character in the matrix
def get_position(matrix, char):
    for i, row in enumerate(matrix):
        for j, c in enumerate(row):
            if c == char:
                return i, j

# Function to preprocess the message (remove spaces, combine double
letters)
def preprocess_message(message):
    message = message.replace(' ', '').lower()

    # Combine double letters by inserting 'x' between them
    processed_message = []
    i = 0
    while i < len(message):
        if i + 1 < len(message) and message[i] == message[i + 1]:
            processed_message.append(message[i] + 'x')
            i += 1
        else:
            processed_message.append(message[i])
        i += 1

    # If the length of the message is odd, append an 'x' at the end
    if len(processed_message) % 2 != 0:
        processed_message.append('x')

    return processed_message

# Function to encrypt the message using the Playfair cipher
def encrypt(message, matrix):
    message = preprocess_message(message)
    encrypted_message = []

    for i in range(0, len(message), 2):
        row1, col1 = get_position(matrix, message[i])
        row2, col2 = get_position(matrix, message[i + 1])

        # If both letters are in the same row, take the letters to the
immediate right
        if row1 == row2:
            encrypted_message.append(matrix[row1][(col1 + 1) % 5])
            encrypted_message.append(matrix[row2][(col2 + 1) % 5])
        # If both letters are in the same column, take the letters
immediately below
        elif col1 == col2:
            encrypted_message.append(matrix[(row1 + 1) % 5][col1])
            encrypted_message.append(matrix[(row2 + 1) % 5][col2])
        # If neither of the above, form a rectangle and swap columns
        else:
            encrypted_message.append(matrix[row1][col2])
            encrypted_message.append(matrix[row2][col1])

    return ''.join(encrypted_message)
```

```
# Function to decrypt the message using the Playfair cipher
def decrypt(message, matrix):
    message = preprocess_message(message)
    decrypted_message = []

    for i in range(0, len(message), 2):
        row1, col1 = get_position(matrix, message[i])
        row2, col2 = get_position(matrix, message[i + 1])

        # If both letters are in the same row, take the letters to the
immediate left
        if row1 == row2:
            decrypted_message.append(matrix[row1][(col1 - 1) % 5])
            decrypted_message.append(matrix[row2][(col2 - 1) % 5])
        # If both letters are in the same column, take the letters
immediately above
        elif col1 == col2:
            decrypted_message.append(matrix[(row1 - 1) % 5][col1])
            decrypted_message.append(matrix[(row2 - 1) % 5][col2])
        # If neither of the above, form a rectangle and swap columns
        else:
            decrypted_message.append(matrix[row1][col2])
            decrypted_message.append(matrix[row2][col1])

    return ''.join(decrypted_message).replace('x', '')

# Main code to demonstrate encryption and decryption
def main():
    key = input("Enter the key for Playfair cipher: ")
    message = input("Enter the message to encrypt: ")

    matrix = create_playfair_matrix(key)
    print("Playfair Cipher Matrix:")
    for row in matrix:
        print(row)

    encrypted_message = encrypt(message, matrix)
    print("Encrypted Message:", encrypted_message)

    decrypted_message = decrypt(encrypted_message, matrix)
    print("Decrypted Message:", decrypted_message)

if __name__ == "__main__":
    main()
```

Output:-



```
Enter the key for Playfair cipher: ncle
Enter the message to encrypt: lcbwebw
Playfair Cipher Matrix:
['n', 'c', 'l', 'e', 'a']
['b', 'd', 'f', 'g', 'h']
['i', 'k', 'm', 'o', 'p']
['q', 'r', 's', 't', 'u']
['v', 'w', 'x', 'y', 'z']
Encrypted Message: eldvngxy
Decrypted Message: lcbwebw
```