|  |  |
| --- | --- |
| Name: | Prerna Sunil Jadhav |
| Sap Id: | 60004220127 |
| Class: | T. Y. B. Tech (Computer Engineering) |
| Course: | Information Security Laboratory |
| Course Code: | DJ19CEL603 |
| Experiment No.: | 01 |

**AIM:** Study and Implement Playfair Cipher.

**ENCRYPTION:**

**CODE:**

def toLowerCase(text):

    return text.lower()

def removeSpaces(text):

    newText = ""

    for i in text:

        if i == " ":

            continue

        else:

            newText = newText + i

    return newText

def Diagraph(text):

    Diagraph = []

    group = 0

    for i in range(2, len(text), 2):

        Diagraph.append(text[group:i])

        group = i

    Diagraph.append(text[group:])

    return Diagraph

def FillerLetter(text):

    k = len(text)

    if k % 2 == 0:

        for i in range(0, k, 2):

            if text[i] == text[i+1]:

                new\_word = text[0:i+1] + str('x') + text[i+1:]

                new\_word = FillerLetter(new\_word)

                break

            else:

                new\_word = text

    else:

        for i in range(0, k-1, 2):

            if text[i] == text[i+1]:

                new\_word = text[0:i+1] + str('x') + text[i+1:]

                new\_word = FillerLetter(new\_word)

                break

            else:

                new\_word = text

    return new\_word

list1 = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'k', 'l', 'm',

        'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z']

def generateKeyTable(word, list1):

    key\_letters = []

    for i in word:

        if i not in key\_letters:

            key\_letters.append(i)

    compElements = []

    for i in key\_letters:

        if i not in compElements:

            compElements.append(i)

    for i in list1:

        if i not in compElements:

            compElements.append(i)

    matrix = []

    while compElements != []:

        matrix.append(compElements[:5])

        compElements = compElements[5:]

    return matrix

def search(mat, element):

    for i in range(5):

        for j in range(5):

            if(mat[i][j] == element):

                return i, j

def encrypt\_RowRule(matr, e1r, e1c, e2r, e2c):

    char1 = ''

    if e1c == 4:

        char1 = matr[e1r][0]

    else:

        char1 = matr[e1r][e1c+1]

    char2 = ''

    if e2c == 4:

        char2 = matr[e2r][0]

    else:

        char2 = matr[e2r][e2c+1]

    return char1, char2

def encrypt\_ColumnRule(matr, e1r, e1c, e2r, e2c):

    char1 = ''

    if e1r == 4:

        char1 = matr[0][e1c]

    else:

        char1 = matr[e1r+1][e1c]

    char2 = ''

    if e2r == 4:

        char2 = matr[0][e2c]

    else:

        char2 = matr[e2r+1][e2c]

    return char1, char2

def encrypt\_RectangleRule(matr, e1r, e1c, e2r, e2c):

    char1 = ''

    char1 = matr[e1r][e2c]

    char2 = ''

    char2 = matr[e2r][e1c]

    return char1, char2

def encryptByPlayfairCipher(Matrix, plainList):

    CipherText = []

    for i in range(0, len(plainList)):

        c1 = 0

        c2 = 0

        ele1\_x, ele1\_y = search(Matrix, plainList[i][0])

        ele2\_x, ele2\_y = search(Matrix, plainList[i][1])

        if ele1\_x == ele2\_x:

            c1, c2 = encrypt\_RowRule(Matrix, ele1\_x, ele1\_y, ele2\_x, ele2\_y)

            # Get 2 letter cipherText

        elif ele1\_y == ele2\_y:

            c1, c2 = encrypt\_ColumnRule(Matrix, ele1\_x, ele1\_y, ele2\_x, ele2\_y)

        else:

            c1, c2 = encrypt\_RectangleRule(

                Matrix, ele1\_x, ele1\_y, ele2\_x, ele2\_y)

        cipher = c1 + c2

        CipherText.append(cipher)

    return CipherText

text\_Plain = 'Hello world'

text\_Plain = removeSpaces(toLowerCase(text\_Plain))

PlainTextList = Diagraph(FillerLetter(text\_Plain))

if len(PlainTextList[-1]) != 2:

    PlainTextList[-1] = PlainTextList[-1]+'z'

key = "Playfair"

print("Key text:", key)

key = toLowerCase(key)

Matrix = generateKeyTable(key, list1)

print("Plain Text:", text\_Plain)

CipherList = encryptByPlayfairCipher(Matrix, PlainTextList)

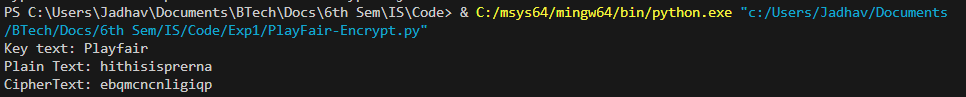
CipherText = ""

for i in CipherList:

    CipherText += i

print("CipherText:", CipherText)

**OUTPUT:**



**DECRYPTION:**

**CODE:**

def toLowerCase(text):

    return text.lower()

def removeSpaces(text):

    newText = ""

    for i in text:

        if i == " ":

            continue

        else:

            newText = newText + i

    return newText

def Diagraph(text):

    Diagraph = []

    group = 0

    for i in range(2, len(text), 2):

        Diagraph.append(text[group:i])

        group = i

    Diagraph.append(text[group:])

    return Diagraph

def FillerLetter(text):

    k = len(text)

    if k % 2 == 0:

        for i in range(0, k, 2):

            if text[i] == text[i+1]:

                new\_word = text[0:i+1] + str('x') + text[i+1:]

                new\_word = FillerLetter(new\_word)

                break

            else:

                new\_word = text

    else:

        for i in range(0, k-1, 2):

            if text[i] == text[i+1]:

                new\_word = text[0:i+1] + str('x') + text[i+1:]

                new\_word = FillerLetter(new\_word)

                break

            else:

                new\_word = text

    return new\_word

list1 = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'k', 'l', 'm',

        'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z']

def generateKeyTable(word, list1):

    key\_letters = []

    for i in word:

        if i not in key\_letters:

            key\_letters.append(i)

    compElements = []

    for i in key\_letters:

        if i not in compElements:

            compElements.append(i)

    for i in list1:

        if i not in compElements:

            compElements.append(i)

    matrix = []

    while compElements != []:

        matrix.append(compElements[:5])

        compElements = compElements[5:]

    return matrix

def search(mat, element):

    for i in range(5):

        for j in range(5):

            if(mat[i][j] == element):

                return i, j

def decrypt\_RowRule(matr, e1r, e1c, e2r, e2c):

    char1 = ''

    if e1c == 0:

        char1 = matr[e1r][4]

    else:

        char1 = matr[e1r][e1c-1]

    char2 = ''

    if e2c == 0:

        char2 = matr[e2r][4]

    else:

        char2 = matr[e2r][e2c-1]

    return char1, char2

def decrypt\_ColumnRule(matr, e1r, e1c, e2r, e2c):

    char1 = ''

    if e1r == 0:

        char1 = matr[4][e1c]

    else:

        char1 = matr[e1r-1][e1c]

    char2 = ''

    if e2r == 0:

        char2 = matr[4][e2c]

    else:

        char2 = matr[e2r-1][e2c]

    return char1, char2

def decrypt\_RectangleRule(matr, e1r, e1c, e2r, e2c):

    char1 = ''

    char1 = matr[e1r][e2c]

    char2 = ''

    char2 = matr[e2r][e1c]

    return char1, char2

def decryptByPlayfairCipher(Matrix, cipherList):

    PlainText = []

    for i in range(0, len(cipherList)):

        c1 = 0

        c2 = 0

        ele1\_x, ele1\_y = search(Matrix, cipherList[i][0])

        ele2\_x, ele2\_y = search(Matrix, cipherList[i][1])

        if ele1\_x == ele2\_x:

            c1, c2 = decrypt\_RowRule(Matrix, ele1\_x, ele1\_y, ele2\_x, ele2\_y)

        elif ele1\_y == ele2\_y:

            c1, c2 = decrypt\_ColumnRule(Matrix, ele1\_x, ele1\_y, ele2\_x, ele2\_y)

        else:

            c1, c2 = decrypt\_RectangleRule(Matrix, ele1\_x, ele1\_y, ele2\_x, ele2\_y)

        plaintext = c1 + c2

        PlainText.append(plaintext)

    return PlainText

text\_Cipher = 'ebqmcncnligiqp'

text\_Cipher = removeSpaces(toLowerCase(text\_Cipher))

# Pad the ciphertext if its length is odd

if len(text\_Cipher) % 2 != 0:

    text\_Cipher += 'x'

CipherTextList = Diagraph(text\_Cipher)

key = "Playfair"

print("Key text:", key)

key = toLowerCase(key)

Matrix = generateKeyTable(key, list1)

print(Matrix)

print("Cipher Text:", text\_Cipher)

PlainTextList = decryptByPlayfairCipher(Matrix, CipherTextList)

PlainText = ""

for i in PlainTextList:

    PlainText += i

print("PlainText:", PlainText)

**OUTPUT:**

