**Experiment 2**

**AIM: Study and implement a program for 5x5 Playfair Cipher to encrypt and decrypt the message.**

**Introduction: The Playfair cipher or Playfair square or Wheatstone–Playfair cipher is a manual symmetric encryption technique and was the first literal digram substitution cipher. The scheme was invented in 1854 by Charles Wheatstone, but bears the name of Lord Playfair for promoting its use.**

**The technique encrypts pairs of letters (bigrams or digrams), instead of single letters as in the simple substitution cipher and rather more complex Vigenère cipher systems then in use. The Playfair cipher is thus significantly harder to break since the frequency analysis used for simple substitution ciphers does not work with it. The frequency analysis of bigrams is possible, but considerably more difficult.**

**Example : For the encryption process let us consider the following example:**

**Key: Monarchy**

**Plaintext: Instruments**

**The Playfair Cipher Encryption Algorithm:**

**The Algorithm consists of 2 steps:**

**1. Generate the key Square(5×5):**

- **The key square is a 5×5 grid of alphabets that acts as the key for encrypting the plaintext. Each of the 25 alphabets must be unique and one letter of the alphabet (usually J) is omitted from the table (as the table can hold only 25 alphabets). If the plaintext contains J, then it is replaced by I.**
- **The initial alphabets in the key square are the unique alphabets of the key in the order in which they appear followed by the remaining letters of the alphabet in order.**

**2. Algorithm to encrypt the plain text: The plaintext is split into pairs of two letters (digraphs). If there is an odd number of letters, a Z is added to the last letter.**

**For example:**

**PlainText: "instruments"**

**After Split: 'in' 'st' 'ru' 'me' 'nt' 'sz**

**Plain Text: "instrumentsz"**

**Encrypted Text: gatlmzclrqtx**

**Source Code:**

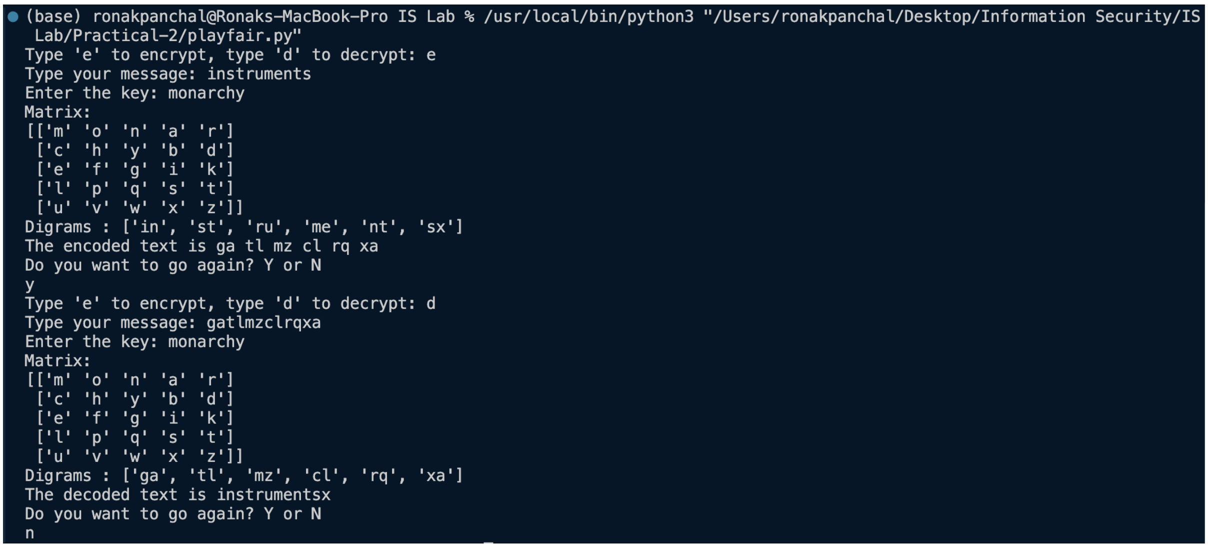
```
import numpy as np
alphabet = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v',
'w', 'x', 'y', 'z']
#Converts the text in to list of two characters
def GroupOfTwo(text):
    res=[]
    group=0
    for i in range(2,len(text),2):
        res.append(text[group:i])
        group=i
    res.append(text[group:])
    return res
#Adds filler character 'x' between two same characters
def Filler(text,direction):
    n = len(text)
    if direction=='e':
        if n % 2 == 0:
            for i in range(0, n, 2):
                if text[i] == text[i+1]:
                    new_word = text[:i+1] + 'x' + text[i+1:]
                    new_word = Filler(new_word)
                    break
                else:
                    new_word = text
        else:
            for i in range(0, n-1, 2):
                if text[i] == text[i+1]:
                    new_word = text[:i+1] + 'x' + text[i+1:]
                    new_word = Filler(new_word)
                    break
```

```
                        else:
                                new_word = text
                if len(new_word)%2!=0:
                        new_word+='x'
        else:
                for i in range(0, n, 2):
                        if text[i] == text[i+1]:
                                new_word = text[:i+1] + 'x' + text[i+1:]
                                new_word = Filler(new_word)
                                break
                        else:
                                new_word = text
                return new_word
#Row rule
def row(matrix,r1,c1,r2,c2,direction):
        char1,char2='',''
                if direction=='e':
                        if c1==4:
                                char1=matrix[r1][0]
                        else:
                                char1=matrix[r1][c1+1]
                        if c2==4:
                                char2=matrix[r2][0]
                        else:
                                char2=matrix[r2][c2+1]
                else:
                        if c1==0:
                                char1=matrix[r1][4]
                        else:
                                char1=matrix[r1][c1-1]
```

```
            if c2==0:
                char2=matrix[r2][4]
            else:
                char2=matrix[r2][c2-1]
    return char1,char2
#Column rule
def column(matrix,r1,c1,r2,c2,direction):
    char1,char2='',''
        if direction=='e':
            if r1==4:
                char1=matrix[0][c1]
            else:
                char1=matrix[r1+1][c1]
            if r2==4:
                char2=matrix[0][c2]
            else:
                char2=matrix[r2+1][c2]
        else:
            if r1==0:
                char1=matrix[4][c1]
            else:
                char1=matrix[r1-1][c1]
            if r2==0:
                char2=matrix[4][c2]
            else:
                char2=matrix[r2-1][c2]
    return char1,char2
#Rectangle rule
def Rectangle(matrix,r1,c1,r2,c2):
    return matrix[r1][c2],matrix[r2][c1]
```

```
#Encryption/Decryption by Playfair Cipher
def playfair(matrix,digrams,direction):
    res=[]
    ans=''
    for i in range(len(digrams)):
        c1=0
        c2=0
        for j in range(5):
            for k in range(5):
                if matrix[j][k]==digrams[i][0]:
                    e1x,e1y=j,k
        for j in range(5):
            for k in range(5):
                if matrix[j][k]==digrams[i][1]:
                    e2x,e2y=j,k
        if e1x==e2x:
            c1,c2=row(matrix,e1x,e1y,e2x,e2y,direction)
        elif e1y==e2y:
            c1,c2=column(matrix,e1x,e1y,e2x,e2y,direction)
        else:
            c1,c2=Rectangle(matrix,e1x,e1y,e2x,e2y)
        cipher=c1+c2
        res.append(cipher)
    for i in res:
        ans+=i
        if direction=='e':
            ans+=' '
    return ans
should_continue=True
while should_continue:
    direction = input("Type 'e' to encrypt, type 'd' to decrypt: ").lower()
    plain_text = input("Type your message: ").lower()
    plain_text=plain_text.replace(" ","")
    key = (input("Enter the key: ")).lower()
    #Create a 5x5 matrix with the unique letters in key and rest of the letter in the alphabet
    key_letter=[]
    composite=[]
    matrix=[]
    for i in key:
        if i not in key_letter:
            key_letter.append(i)
    for i in key_letter:
        if i not in composite:
            composite.append(i)
    for i in alphabet:
        if i not in composite:
            composite.append(i)
    for i in range(0,25,5):
        matrix.append(composite[i:i+5])
    print("Matrix: ")
    print(np.matrix(matrix))
    digrams=GroupOfTwo(Filler(plain_text,direction))
    print(f"Digrams : {digrams}")
    result=playfair(matrix,digrams,direction)
    if direction == 'e':
        print(f"The encoded text is {result}")
    elif direction == 'd':
        print(f"The decoded text is {result}")
    repeat = input("Do you want to go again? Y or N\n").lower()
    if repeat == 'n':
        should_continue = False
```

**Output:**



```
● (base) ronakpanchal@Ronaks-MacBook-Pro IS Lab % /usr/local/bin/python3 "/Users/ronakpanchal/Desktop/Information Security/IS
   Lab/Practical-2/playfair.py"
  Type 'e' to encrypt, type 'd' to decrypt: e
  Type your message: instruments
  Enter the key: monarchy
  Matrix:
  [['m' 'o' 'n' 'a' 'r']
   ['c' 'h' 'y' 'b' 'd']
   ['e' 'f' 'g' 'i' 'k']
   ['l' 'p' 'q' 's' 't']
   ['u' 'v' 'w' 'x' 'z']]
  Digrams : ['in', 'st', 'ru', 'me', 'nt', 'sx']
  The encoded text is ga tl mz cl rq xa
  Do you want to go again? Y or N
  y
  Type 'e' to encrypt, type 'd' to decrypt: d
  Type your message: gatlmzclrqxa
  Enter the key: monarchy
  Matrix:
  [['m' 'o' 'n' 'a' 'r']
   ['c' 'h' 'y' 'b' 'd']
   ['e' 'f' 'g' 'i' 'k']
   ['l' 'p' 'q' 's' 't']
   ['u' 'v' 'w' 'x' 'z']]
  Digrams : ['ga', 'tl', 'mz', 'cl', 'rq', 'xa']
  The decoded text is instrumentsx
  Do you want to go again? Y or N
  n
```

**Time Complexity:** **$O(n^2)$ for both Encryption and Decryption (n is length of plain text).**

**Space Complexity:** **O(n) for both Encryption and Decryption.**

## Revised

**Logic:**

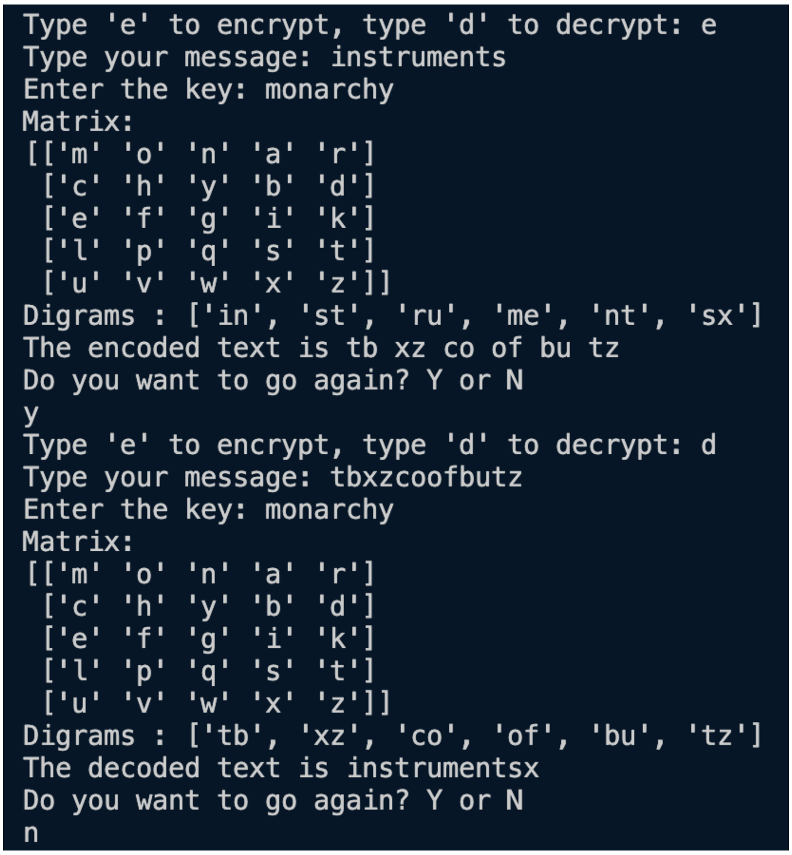
**If the Letter are in the same row, then we will go 1 row down and vice versa. Similarly, if the letters are in the same column, then we will go to 1 column right and vice versa. And if the letters are in the different co-ordinate, then we will go in diagonal side**

**Source Code:**

```
import numpy as np



alphabet = ['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']



#Converts the text in to list of two characters

def GroupOfTwo(text):

    res=[]

    group=0

    for i in range(2,len(text),2):

        res.append(text[group:i])

        group=i

    res.append(text[group:])

    return res

#Adds filler character 'x' between two same characters

def Filler(text,direction):

    n = len(text)

    if direction=='e':

        if n % 2 == 0:

            for i in range(0, n, 2):

                if text[i] == text[i+1]:

                    new_word = text[:i+1] + 'x' + text[i+1:]
                    new_word = Filler(new_word)
                    break
            else:
                new_word = text
        else:
            for i in range(0, n-1, 2):
                if text[i] == text[i+1]:
                    new_word = text[:i+1] + 'x' + text[i+1:]
                    new_word = Filler(new_word)
                    break
                else:
                    new_word = text
        if len(new_word)%2!=0:
            new_word+='x'
    else:
        for i in range(0, n, 2):
            if text[i] == text[i+1]:
                new_word = text[:i+1] + 'x' + text[i+1:]
                new_word = Filler(new_word)
                break
            else:
                new_word = text
    return new_word


#New Row rule
def row(matrix,r1,c1,r2,c2,direction):
    char1,char2='',''
    if direction=='e':

        char1=matrix[(r1+1)%5][c1]

        char2=matrix[(r2+1)%5][c2]

    else:

        char1=matrix[(r1-1)%5][c1]

        char2=matrix[(r2-1)%5][c2]

    return char1,char2


#New Column rule

def column(matrix,r1,c1,r2,c2,direction):

    char1,char2=","

    if direction=='e':

        char1=matrix[r1][(c1+1)%5]

        char2=matrix[r2][(c2+1)%5]

    else:

        char1=matrix[r1][(c1-1)%5]

        char2=matrix[r2][(c2-1)%5]

return char1,char2

#New Rectangle rule

def Rectangle(matrix,r1,c1,r2,c2,direction):

    if direction=="e":

        return matrix[(r1+1)%5][(c1+1)%5],matrix[(r2+1)%5][(c2+1)%5]

    else:

        return matrix[(r1-1)%5][(c1-1)%5],matrix[(r2-1)%5][(c2-1)%5]


#Encryption/Decryption by Revised Playfair Cipher

def playfair(matrix,digrams,direction):
    res=[]
    ans=''
    for i in range(len(digrams)):
        c1=0
        c2=0
        for j in range(5):
            for k in range(5):
                if matrix[j][k]==digrams[i][0]:
                    e1x,e1y=j,k
        for j in range(5):
            for k in range(5):
                if matrix[j][k]==digrams[i][1]:
                    e2x,e2y=j,k
        if e1x==e2x:
            c1,c2=row(matrix,e1x,e1y,e2x,e2y,direction)
        elif e1y==e2y:
            c1,c2=column(matrix,e1x,e1y,e2x,e2y,direction)
        else:
            c1,c2=Rectangle(matrix,e1x,e1y,e2x,e2y,direction)
        cipher=c1+c2
        res.append(cipher)
        for i in res:
            ans+=i
            if direction=='e':
                ans+=' '
        return ans
should_continue=True
```

```
while should_continue:
    direction = input("Type 'e' to encrypt, type 'd' to decrypt: ").lower()
    plain_text = input("Type your message: ").lower()
    plain_text=plain_text.replace(" ","")
    key = (input("Enter the key: ")).lower()
    #Create a 5x5 matrix with the unique letters in key and rest of the letter in the alphabet
    key_letter=[]
    composite=[]
    matrix=[]
    for i in key:
        if i not in key_letter:
            key_letter.append(i)
    for i in key_letter:
        if i not in composite:
            composite.append(i)
    for i in alphabet:
        if i not in composite:
            composite.append(i)
    for i in range(0,25,5):
        matrix.append(composite[i:i+5])
    print("Matrix: ")
    print(np.matrix(matrix))
    digrams=GroupOfTwo(Filler(plain_text,direction))
    print(f"Digrams : {digrams}")
    result=playfair(matrix,digrams,direction)
    if direction == 'e':
        print(f"The encoded text is {result}")
    elif direction == 'd':
        print(f"The decoded text is {result}")
    repeat = input("Do you want to go again? Y or N\n").lower()
    if repeat == 'n':
        should_continue = False
```

**Output:**



```
Type 'e' to encrypt, type 'd' to decrypt: e
Type your message: instruments
Enter the key: monarchy
Matrix:
[['m' 'o' 'n' 'a' 'r']
 ['c' 'h' 'y' 'b' 'd']
 ['e' 'f' 'g' 'i' 'k']
 ['l' 'p' 'q' 's' 't']
 ['u' 'v' 'w' 'x' 'z']]
Digrams : ['in', 'st', 'ru', 'me', 'nt', 'sx']
The encoded text is tb xz co of bu tz
Do you want to go again? Y or N
y
Type 'e' to encrypt, type 'd' to decrypt: d
Type your message: tbxzcoofbutz
Enter the key: monarchy
Matrix:
[['m' 'o' 'n' 'a' 'r']
 ['c' 'h' 'y' 'b' 'd']
 ['e' 'f' 'g' 'i' 'k']
 ['l' 'p' 'q' 's' 't']
 ['u' 'v' 'w' 'x' 'z']]
Digrams : ['tb', 'xz', 'co', 'of', 'bu', 'tz']
The decoded text is instrumentsx
Do you want to go again? Y or N
n
```

**Time Complexity:** **$O(n^2)$ for both Encryption and Decryption (n is length of plaintext).**

**Space Complexity:** **O(n) for both Encryption and Decryption.**