Name : Harshal Kodgire
Batch : B1
Subject : CNS lab
Topic : Assignment 3

Aim : To implement Playfair Cipher

Theory :
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. Pair cannot be made with the same letter. Break the letter in single and add a bogus letter to the previous letter. If both the letters are in the same column: Take the letter below each one. If both the letters are in the same row: Take the letter to the right of each one. If neither of the above rules is true: Form a rectangle with the two letters and take the letters on the horizontal opposite corner of the rectangle

Code :

```
#include<bits/stdc++.h>
using namespace std;

int main()
{
    string key;
    string cipherText;
    string plainText;
    char matrix[5][5];
    int freqArr[26] = {0};
    vector<pair<int,int>> location(26,{-1,-1});

    cout<<"\n Enter key : ";
    getline(cin,key);

    cout<<"\n Enter plain text : ";
    getline(cin,plainText);

    // Removing spaces from key
    string temp = "";
    for(int i=0;i<key.size();i++)
    {
        if(key[i]!=' ')
```

```
        temp += key[i];
    }
    key = temp;

    for(int i=0;i<key.size();i++)
    {
        if(key[i]>=65 && key[i]<=90)
        key[i] += 32;
    }

    // Removing spaces from plain text and converting in small
character
    string temp2 = "";
    for(int i=0;i<plainText.size();i++)
    {
        if(plainText[i]!=' ')
        temp2 += plainText[i];
    }
    plainText = temp2;

    for(int i=0;i<plainText.size();i++)
    {
        if(plainText[i]>=65 && plainText[i]<=90)
        plainText[i] += 32;
    }
    cipherText = plainText;

    int row = 0,col = 0;

    for(int i=0;i<key.size();i++)
    {
        row = row % 5;
        col = col % 5;

        if(freqArr[key[i]-'a']==0)
        {
            if((key[i]=='j' || key[i]=='i') && (freqArr[8] ||
freqArr[9]))
            continue;

            matrix[row][col] = key[i];
            col++;
            freqArr[key[i]-'a'] = 1;
```

```
        }

        if(col==5)
        {
            col = 0;
            row++;
        }
    }

    for(int i=0;i<26;i++)
    {
        if(freqArr[i]==0)
        {
            if((i==8 && freqArr[9]==1) || (i==9 && freqArr[8]==1)){
            continue;
            }
            matrix[row][col] = 'a'+i;
            col++;
            freqArr[i] = 1;
        }
    }

    cout<<endl;
    for(int i=0;i<5;i++)
    {
        for(int j=0;j<5;j++)
        {
            cout<<matrix[i][j]<<" ";
            location[matrix[i][j]-'a'] = {i,j};
        }
        cout<<endl;
    }

    char dummyChar = matrix[5][5];

    cout<<"Plain Text : "<<plainText<<endl;
    cout<<"Key : "<<key<<endl;

    for(int i=0;i<plainText.size();)
    {
        cout<<i<<endl;
        if(i+1<plainText.size())
        {
```

```
            if(plainText[i]==plainText[i+1])
            {
                // Taking last char of matrix as dummy
                cipherText[i] =
matrix[location[plainText[i]].first][4];
                i = i + 1;
            }
            else
            {
                // check if both have same row
                if(location[plainText[i]-'a'].first ==
location[plainText[i+1]-'a'].first)
                {
                    cipherText[i] =
matrix[location[plainText[i]-'a'].first][(location[plainText[i]-'a'].se
cond+1)%5];
                    cipherText[i+1] =
matrix[location[plainText[i+1]-'a'].first][(location[plainText[i+1]-'a'
].second+1)%5];
                }

                // check if both have same column
                else if(location[plainText[i]-'a'].second ==
location[plainText[i+1]-'a'].second)
                {
                    cipherText[i] =
matrix[(location[plainText[i]-'a'].first+1)%5][location[plainText[i]-'a
'].second];
                    cipherText[i+1] =
matrix[(location[plainText[i+1]-'a'].first+1)%5][location[plainText[i+1
]-'a'].second];
                }

                // Interchange columns
                else
                {
                    cipherText[i] =
matrix[location[plainText[i]-'a'].first][location[plainText[i+1]-'a'].s
econd];
                    cipherText[i+1] =
matrix[location[plainText[i+1]-'a'].first][location[plainText[i]-'a'].s
econd];
                }
```

```
                i = i + 2;
            }
        }
        else
        {
            // Taking last char of matrix as dummy
            cipherText[i] = matrix[location[key[i]].first][4];
            i = i + 1;
        }
    }
    cout<<"\n Cipher Text : "<<cipherText<<endl;

    return 0;
}
```

Output :

```
D:\WCE_ENGINEERING\BTECH_SEM1\CNS lab>a.exe

 Enter key : occurence

 Enter plain text : thekeyissecret

o c u r e
n a b d f
g h i k l
m p q s t
v w x y z
Plain Text : thekeyissecret
Key : occurence

 Cipher Text : plrlrzkqtruefz
```