**Cryptography & Network Security**

PRN - 2019BTECS00026

Name - Niraja Vasudev Kulkarni

Batch - B1

**Assignment - 3**

**Playfair Cipher**

* **Objective -**

To implement Playfair Cipher in C

* **Theory -**

The **Playfair cipher** was the first practical digraph substitution cipher. The scheme was invented in **1854**by**Charles Wheatstone** but was named after Lord Playfair who promoted the use of the cipher. In Playfair cipher unlike traditional cipher we encrypt a pair of alphabets(digraphs) instead of a single alphabet.  
Procedure -

1. Take the plaintext & the key as input
2. Arrange the key in 5\*5 matrix without repetition of characters. Fill rest of the matrix with remaining alphabets serially (Skip I/J)
3. Prepare chunks of size 2 in plaintext. If 2 characters in the chunk are same , put a dummy character.
4. Make sure the plaintext length is even otherwise put a dummy character in the end
5. For each chunk , encrypt the text by using playfair matrix with the help of following rules :
6. If both of them are in same row , take the next characters in the same row (consider wraparound)
7. If both of them are in same column , take the next characters in the same column (consider wraparound)
8. Otherwise take their intersection

**Code –**

#include <bits/stdc++.h>

using namespace std;

int RemoveSpaces(char arr[], int len)

{

    int i, count = 0;

    for (i = 0; i < len; i++)

        if (arr[i] != ' ')

            arr[count++] = arr[i];

    arr[count] = '\0';

    return count;

}

void GenerateKeyTable(char keyarr[5], int keylen, char keyTable[5][5])

{

    int i, j, k, flag = 0;

    int map[26] = {0};

    for (i = 0; i < keylen; i++)

    {

        if (keyarr[i] != 'J')

            map[keyarr[i] - 65] = 2;

    }

    map['J' - 65] = 1;

    i = 0;

    j = 0;

    for (k = 0; k < keylen; k++)

    {

        if (map[keyarr[k] - 65] == 2)

        {

            map[keyarr[k] - 65] -= 1;

            keyTable[i][j] = keyarr[k];

            j++;

            if (j == 5)

            {

                i++;

                j = 0;

            }

        }

    }

    for (k = 0; k < 26; k++)

    {

        if (map[k] == 0)

        {

            keyTable[i][j] = (char)(k + 65);

            j++;

            if (j == 5)

            {

                i++;

                j = 0;

            }

        }

    }

}

void search(char keyTable[5][5], char a, char b, int arr[])

{

    int i, j;

    if (a == 'J')

        a = 'I';

    else if (b == 'J')

        b = 'I';

    for (i = 0; i < 5; i++)

    {

        for (j = 0; j < 5; j++)

        {

            if (keyTable[i][j] == a)

            {

                arr[0] = i;

                arr[1] = j;

            }

            else if (keyTable[i][j] == b)

            {

                arr[2] = i;

                arr[3] = j;

            }

        }

    }

}

int mod5(int a)

{

    if (a < 0)

        a += 5;

    return (a % 5);

}

int Prepare(char arr[], int len)

{

    string res;

    int i = 0;

    while (i < len - 1)

    {

        res.push\_back(arr[i]);

        if (arr[i] == arr[i + 1])

        {

            res.push\_back('X');

            i++;

        }

        else

        {

            res.push\_back(arr[i + 1]);

            i += 2;

        }

    }

    len = res.size();

    strcpy(arr, res.c\_str());

    if (len % 2 != 0)

    {

        arr[len++] = 'X';

        arr[len] = '\0';

    }

    return len;

}

void Encrypt(char textarr[], char keyTable[5][5], int textlen)

{

    int i, arr[4];

    for (i = 0; i < textlen; i += 2)

    {

        search(keyTable, textarr[i], textarr[i + 1], arr);

        if (arr[0] == arr[2])

        {

            textarr[i] = keyTable[arr[0]][mod5(arr[1] + 1)];

            textarr[i + 1] = keyTable[arr[0]][mod5(arr[3] + 1)];

        }

        else if (arr[1] == arr[3])

        {

            textarr[i] = keyTable[mod5(arr[0] + 1)][arr[1]];

            textarr[i + 1] = keyTable[mod5(arr[2] + 1)][arr[1]];

        }

        else

        {

            textarr[i] = keyTable[arr[0]][arr[3]];

            textarr[i + 1] = keyTable[arr[2]][arr[1]];

        }

    }

}

void Decrypt(char textarr[], char keyTable[5][5], int textlen)

{

    string res;

    int i, arr[4];

    for (i = 0; i < textlen; i += 2)

    {

        search(keyTable, textarr[i], textarr[i + 1], arr);

        if (arr[0] == arr[2])

        {

            textarr[i] = keyTable[arr[0]][mod5(arr[1] - 1)];

            textarr[i + 1] = keyTable[arr[0]][mod5(arr[3] - 1)];

        }

        else if (arr[1] == arr[3])

        {

            textarr[i] = keyTable[mod5(arr[0] - 1)][arr[1]];

            textarr[i + 1] = keyTable[mod5(arr[2] - 1)][arr[1]];

        }

        else

        {

            textarr[i] = keyTable[arr[0]][arr[3]];

            textarr[i + 1] = keyTable[arr[2]][arr[1]];

        }

    }

    for (i = 0; i < textlen; i++)

    {

        if(textarr[i]!='X') res.push\_back(textarr[i]);

    }

    strcpy(textarr, res.c\_str());

}

string PlayfairEncrypt(char textarr[], char keyarr[],int flg)

{

    char keyTable[5][5];

    int i,j;

    string result;

    int keylen = RemoveSpaces(keyarr, strlen(keyarr));

    int textlen = RemoveSpaces(textarr, strlen(textarr));

    textlen = Prepare(textarr, textlen);

    GenerateKeyTable(keyarr, keylen, keyTable);

    if(flg==1) {

        for(i = 0; i < 5; i++) {

            for(j = 0; j < 5; j++) {

                cout<<keyTable[ i ] [ j ]<<" ";

            }

            cout<<"\n";

        }

    }

    Encrypt(textarr, keyTable, textlen);

    result = textarr;

    return result;

}

string PlayfairDecrypt(char textarr[], char keyarr[])

{

    char keyTable[5][5];

    string result;

    int keylen = strlen(keyarr);

    int textlen = strlen(textarr);

    GenerateKeyTable(keyarr, keylen, keyTable);

    Decrypt(textarr, keyTable, textlen);

    result = textarr;

    return result;

}

int main()

{

    int option,flg=0;

    string key, text, cipherText;

    char keyarr[30], textarr[30], cipher[30];

    cout << "\nGive input through:\n1)Console\n2)File\n-->";

    cin >> option;

    cin.ignore();

    cout << "Enter key: ";

    getline(cin, key);

    strcpy(keyarr, key.c\_str());

    switch (option)

    {

        case 1:

            cout << "Enter text: ";

            flg=1;

            break;

        case 2:

            freopen("input.txt", "r", stdin);

            freopen("output.txt", "w", stdout);

            flg=2;

            break;

    }

    getline(cin, text);

    strcpy(textarr, text.c\_str());

    cipherText = PlayfairEncrypt(textarr, keyarr,flg);

    cout << "Cipher Text: " << cipherText << endl;

    strcpy(cipher, cipherText.c\_str());

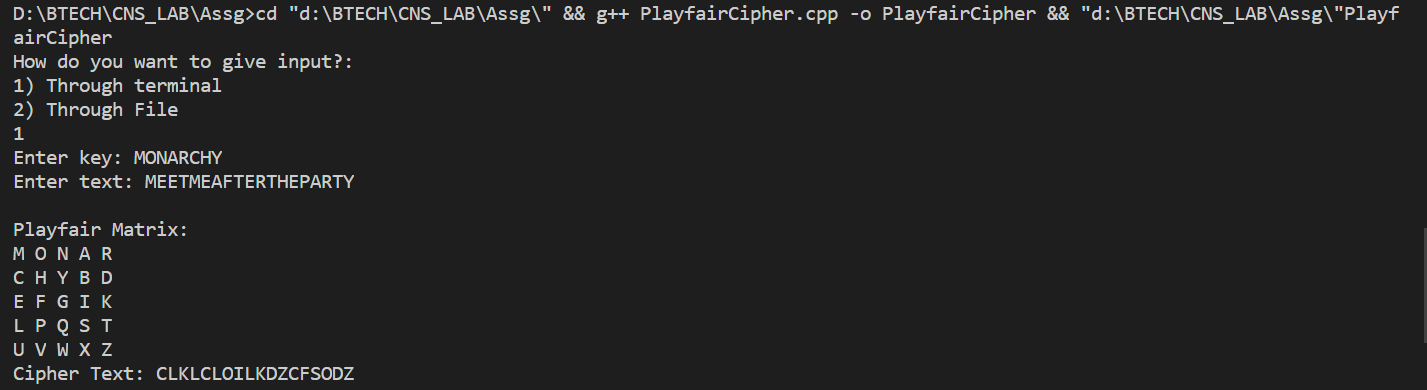
    cout << "Deciphered Text: " << PlayfairDecrypt(cipher, keyarr);

    return 0;

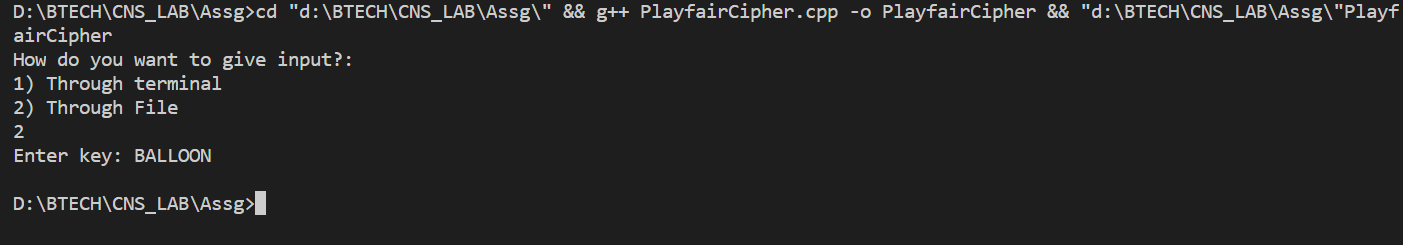
}

**Output –**

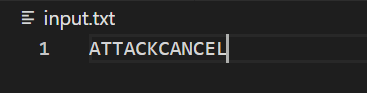
Sample output 1 –



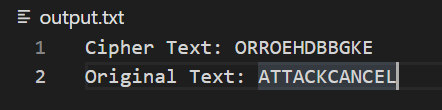
Sample output 2 -



Input file



Output file



**Conclusion –**

Playfair cipher is a polyalphabetic substitution cipher . Due to its easy decrption , it is not used in the modern encryption.