EXP NO:2
DATE:

PLAYFAIR CIPHER

AIM:

 To implement an encryption algorithm using Playfair Cipher technique.

ALGORITHM:

 Step 1: Get the plaintext input from the user.
 Step 2: Generate the Playfair Cipher key table based on a given keyword.
 Step 3: Preprocess the plaintext by removing any characters that aren't letters and converting it to uppercase.
 Step 4: If necessary, handle repeated letters in the plaintext by inserting a filler character or breaking them apart with another letter.
 Step 5: Divide the preprocessed plaintext into pairs of two letters. If the number of letters is odd, add a filler character to make it even.
 Step 6: Apply the encryption rules of the Playfair Cipher to each digraph.
 Step 7: Output the encrypted ciphertext formed by the transformed digraphs.

PROGRAM:
```
#include  <stdio.h>
#include <stdlib.h>
#include <string.h>
#define SIZE 30
void toLowerCase(char plain[], int ps)
{
   int i;
   for (i = 0; i < ps; i++) {
     if (plain[i] > 64 && plain[i] < 91)
        plain[i] += 32;
   }
}
int removeSpaces(char* plain, int ps)
{
```

```c
    for (i = 0; i < ps; i++)
        if (plain[i] != ' ')
            plain[count++] = plain[i];
    plain[count] = '\0';
    return count;
}
void generateKeyTable(char key[], int ks, char keyT[5][5])
{
    int i, j, k, flag = 0, *dicty;
    dicty = (int*)calloc(26, sizeof(int));

    for (i = 0; i < ks; i++) {
        if (key[i] != 'j')
            dicty[key[i] - 97] = 2;
    }
    dicty['j' - 97] = 1;
    i = 0;
    j = 0;
    for (k = 0; k < ks; k++) {
        if (dicty[key[k] - 97] == 2) {
            dicty[key[k] - 97] -= 1;
            keyT[i][j] = key[k];
            j++;
            if (j == 5) {
                i++;
                j = 0;
            }
        }
    }
    for (k = 0; k < 26; k++) {
        if (dicty[k] == 0) {
            keyT[i][j] = (char)(k + 97);
            j++;
            if (j == 5) {
                i++;
```

```c
            j = 0;
        }
    }
}
void search(char keyT[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 (keyT[i][j] == a) {
                arr[0] = i;
                arr[1] = j;
            }
            else if (keyT[i][j] == b) {
                arr[2] = i;
                arr[3] = j;
            }
        }
    }
}
int mod5(int a)
{
    if (a < 0)
        a += 5;
    return (a % 5);
}
void decrypt(char str[], char keyT[5][5], int ps)
{
```

```c
    int i, a[4];
    for (i = 0; i < ps; i += 2) {
        search(keyT, str[i], str[i + 1], a);
        if (a[0] == a[2]) {
            str[i] = keyT[a[0]][mod5(a[1] - 1)];
            str[i + 1] = keyT[a[0]][mod5(a[3] - 1)];
        }
        else if (a[1] == a[3]) {
            str[i] = keyT[mod5(a[0] - 1)][a[1]];
            str[i + 1] = keyT[mod5(a[2] - 1)][a[1]];
        }
        else {
            str[i] = keyT[a[0]][a[3]];
            str[i + 1] = keyT[a[2]][a[1]];
        }
    }
}

void decryptByPlayfairCipher(char str[], char key[])
{
    char ps, ks, keyT[5][5];
    ks = strlen(key);
    ks = removeSpaces(key, ks);
    toLowerCase(key, ks);
    ps = strlen(str);
    toLowerCase(str, ps);
    ps = removeSpaces(str, ps);

    generateKeyTable(key, ks, keyT);

    decrypt(str, keyT, ps);
}

int main()
{
```

```
    char str[SIZE], key[SIZE];

     strcpy(key, "Thrisha");
    printf("Key text: %s\n", key);
    strcpy(str, "ulroaliocvrx");
    printf("Plain text: %s\n", str);

        decryptByPlayfairCipher(str, key);

    printf("Deciphered text: %s\n", str);

    return 0;
}
```

OUTPUT:

```
Key text: Ukesh
Plain text: Helloworld
Deciphered text: vhkliyykdt
```

RESULT: