NIS

LAB - 3

ROll NO. : CE146

Name : Shingala Shubham P.

Id NO.: 19CEUOS1569

* AIM : Write a program to implement
  1) PlayFair Cipher
  2) AutoKey Cipher

1) PlayFair Cipher :

```
#include <bits/stdc++.h>
using namespace std;
int keyMatrix[5][5];
class pointer {
    public:
        int x;
        int y;
};

pointer findInMatrix (char a, char b,
        pointer P[]) {
    if (a=='J')
        a = 'I';
    int i, j;
    for (i=0; i<5; i++)
        for (j=0; j<5; j++)
```

```
        if (keyMatrix[i][j] == a) {
            p[0].x = i;
            p[0].y = j;
        }
        else if (keyMatrix[i][j] == b) {
            p[1].x = i;
            p[1].y = j;
        }
    }

string removeSpaces(string str)
{
    str.erase(remove(str.begin(), str.end(),
                     ' '), str.end());
    return str;
}

int mode(int a, int n) {
    int modeN = a % n;
    if (modeN < 0)
        modeN += n;
    return modeN;
}
```

```
void generatematrix(string key) {
    int i, j, k=0, l=key.length();
    char s='A';
    for(i=0; i<5; i++) {
        for(j=0; j<5;) {
            if(k<l) {
                keymatrix[i][j] = key[k];
                k++; j++;
            }
            else {
                if(s=='J')
                    s++
                if(key.find(s)==string::npos)
                {
                    keymatrix[i][j] = s;
                    s++; j++;
                }
                else
                    s++;
            }
        }
    }
}
```

```
string encrypt (string plainText) {
    int l = plainText.length(), i, j;
    string encryptText;
    if (l % 2 != 0)
        plainText [l] = 'z';
    for (i=0 ; i< l; i++) {
        pointer p[2];
        findInMatrix (toupper (plainText[i]),
            toupper (plainText[i+1], p);
        i++;
        if (p[0].x == p[1].x) {
            encryptText += keymatrix [p[0].x]
                [mode(p[0].y+1, 5)];
            encryptText += keymatrix [p[0].x]
                [mode (p[1].y+1, 5)];
        }
        else if (p[0].y == p[1].y) {
            encryptText += keymatrix [mode(
                p[0].x+1, 5)] [p[0].y];
            encryptText += keymatrix [mode(
                p[1].x+1, 5)] [p[0].y];
        }
        else {
            encryptText += keymatrix [p[0].x]
                [p[1].y];
            encryptText += keymatrix [p[1].x]
                [p[0].y];
        }
    }
    return encryptText;
}
```

```
string decrypt ( string cipherText) {
    int l = cipherText.length ( ), i, j;
    string decryptText;
    for (i=0; i<l; i++) {
        pointer P[2];
        Find In matrix (cipherText[i], cipherText
                        [i+1], P);
        i++;
        if (P[0].x == P[1].x) {
            decryptText += key matrix [P[0].x]
                [mode(P[0].y-1, 5)];
            decryptText += key matrix [P[0].x]
                [mode(P[1].y-1, 5)];
        }
        else if (P[0].y == P[1].y) {
            decryptText += key matrix [mode(
                P[0].x-1, 5)] [P[0].y];
            decryptText += key matrix [mode(
                P[1].x-1, 5)] [P[0].y];
        }
        else {
            decryptText += key matrix [P[0].x]
                [P[1].y]
            decryptText += key matrix [P[1].x]
                [P[0].y]
        }
    }
    transform (decryptText.begin(), decryptText.
        end(), decryptText.begin(), ::tolower);
    return decryptText;
}
```

```
int main() {
    string plainText, key = "MONARCHY";
    generateMatrix(key);
    cout << "Plain Text:";
    getline(cin, plainText);
    plainText = removeSpaces(plainText);
    string encryptText = encrypt(plainText);
    cout << "Cipher Text:" << encryptText
        << endl << "DecryptText:" <<
        decrypt(encryptText);
}
```

→ Test - case - 1 ;

Input : Plain Text : Playfair

Output : Cipher Text : QPNBIOKA
Decrypt Text : Playfair.

→ Test case - 2 :

Input : Plain Text : Shubham X Shingaid

Output : Cipher Text : PBXCBOAUPBQAINSM
Decrypt Text : shubhamxshingaid

2) Auto Key cipher :

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

string encrypt(int key, string plainText){
    string cipherText;
    int i, textmap;
    transform(plainText.begin(), plainText.
        end(), plainText.begin(), ::toupper);
    for(i=0; i<plainText.length(); i++){
        textmap = plainText[i] - 'A';
        cipherText += mode(textmap + key, 26)
                    + 'A';
        key = textmap;
    }
    return cipherText;
}

string decrypt(int key, string cipherText){
    string plainText;
    int i, textmap;
    for(i=0; i<cipherText.length(); i++){
        textmap = cipherText[i] - 'A';
        plainText += mode(textmap - key,
            26) + 'a';
        key = plainText[i] - 'a';
    }
    return plainText;
}
```

```
int main() {
    string plainText;
    cout << "Enter plain Text:";
    cin >> plainText;
    int key;
    cout << "Enter key:";
    cin >> key;
    string encryptText = encrypt(
        key, plainText);
    cout << "Encrypt Text:" << encryptText
        << endl << "DecryptText Text:"
        << decrypt (key, encryptText);
    return 0;
}
```

→ Test case - 1 :

Input : cryto key cipher key = 15

Output : Encrypt Text : PUNHYOCA KX WLV
Decrypt Text : cryto key cipher

→ Test case - 2 :

Input : Shubham key = 20

Output : Encrypt Text : MZBVIHM
Decrypt text : Shubham