Name: Karuna Bajirao Randive.

Prn: 2020BTECS00024

Batch: B2

Cryptography & Network Security Lab

Assignment No. 4

Theory:

Playfair Cipher Technique

The Playfair Cipher Technique is a substitution cipher that encrypts pairs of characters (digraphs) from the plaintext using a 5x5 key square matrix.

The matrix is constructed from a keyword, with duplicate letters removed and the keyword letters placed at the beginning.

Encryption involves applying rules based on the positions of the letters within the key square.

If the letters are in the same row, column, or form a rectangle, they are replaced by specific neighbouring letters.

Encryption:

- Divide the plaintext into digraphs.
- Apply rules based on the positions of letters in the key square to replace each digraph.

Decryption:

- Divide the ciphertext into digraphs.
- Apply the rules in reverse to each digraph to retrieve the original plaintext.

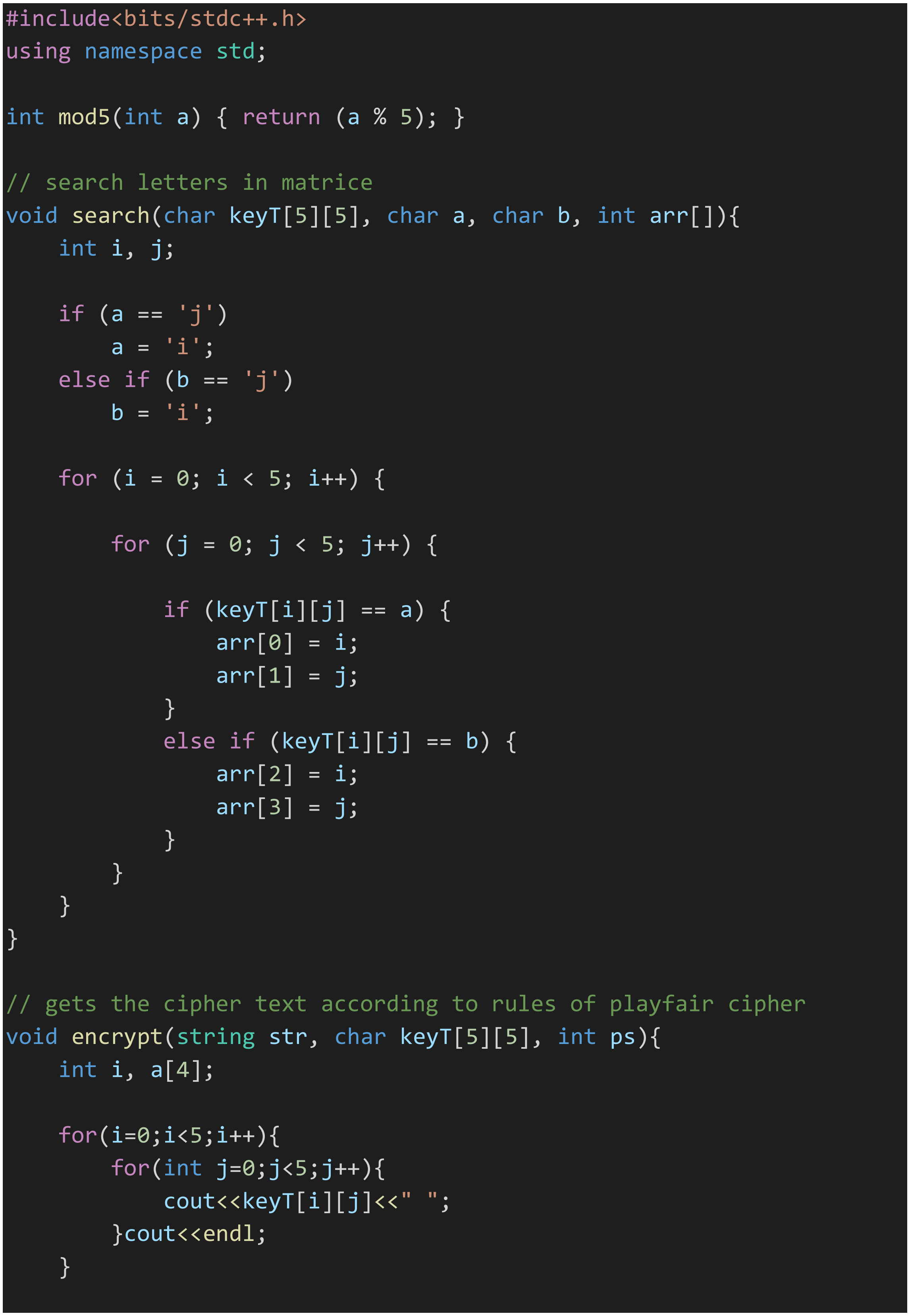
Advantages:

- Enhanced security due to digraphs and key square usage.
- Reduces susceptibility to frequency analysis.
- Key square generation is straightforward using a keyword.

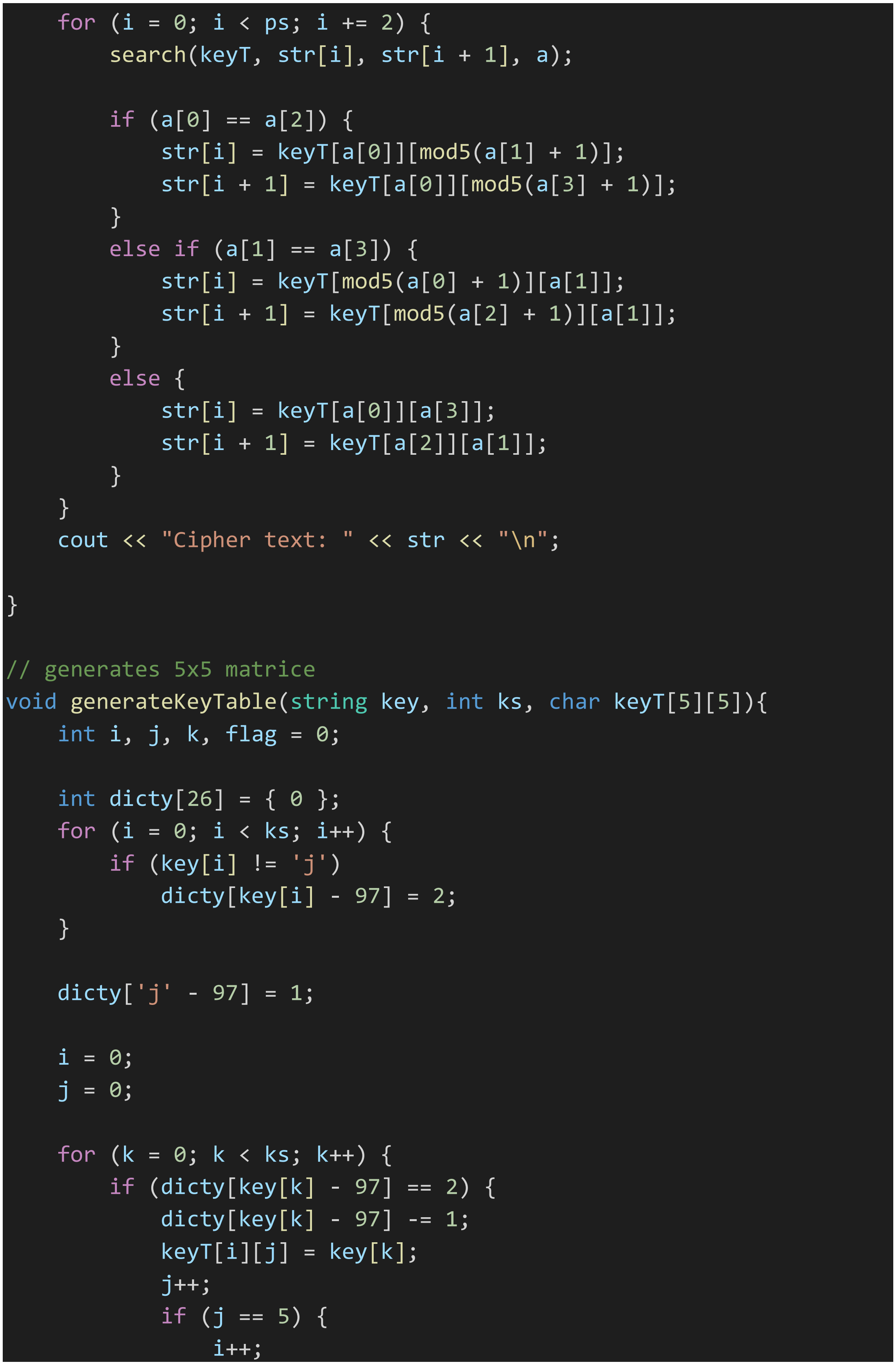
Disadvantages:

- Complexity increases with handling various cases (same row, column, rectangle).
- Security depends on the keyword and arrangement of the key square.

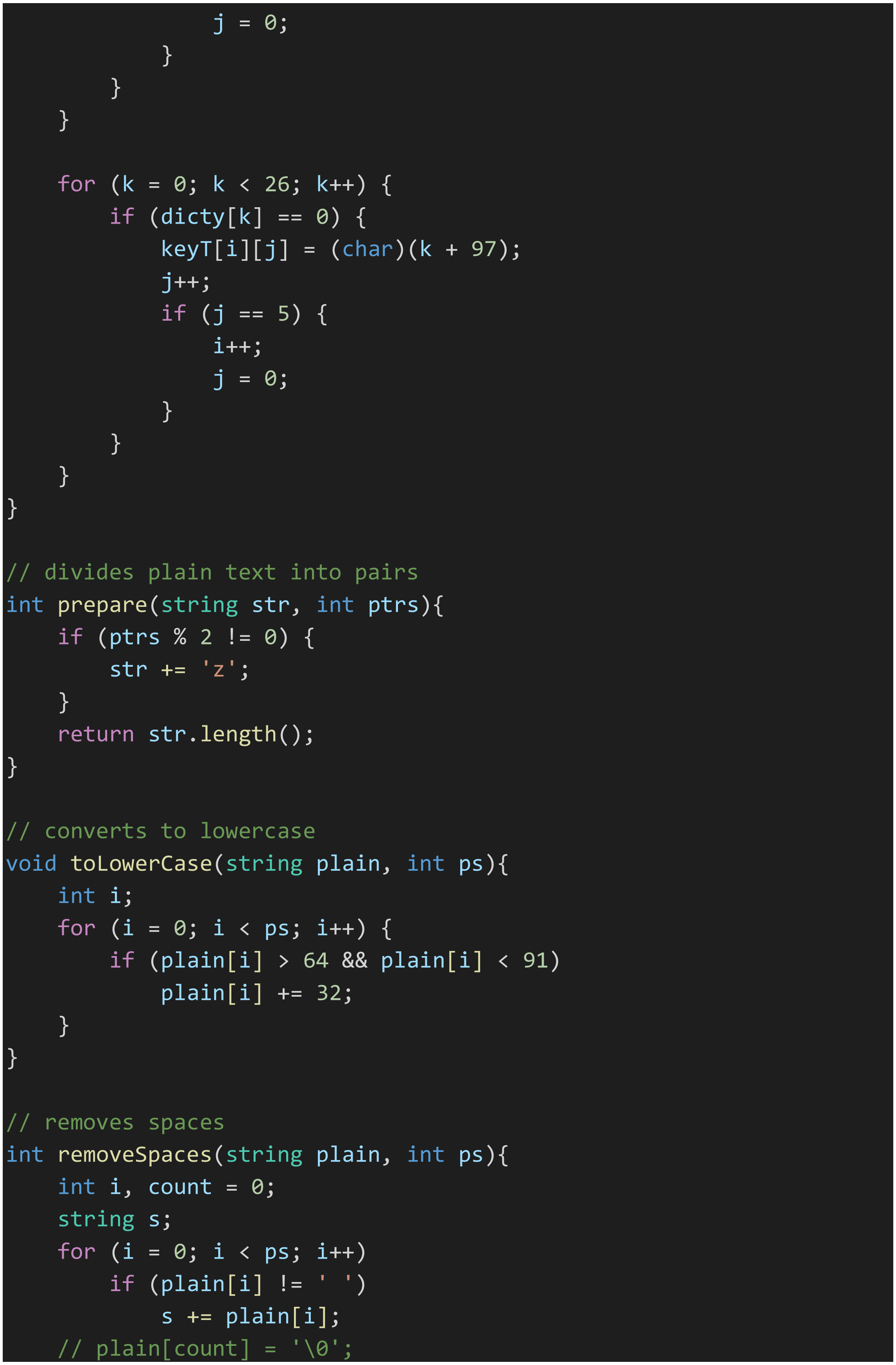
Playfair Cipher:



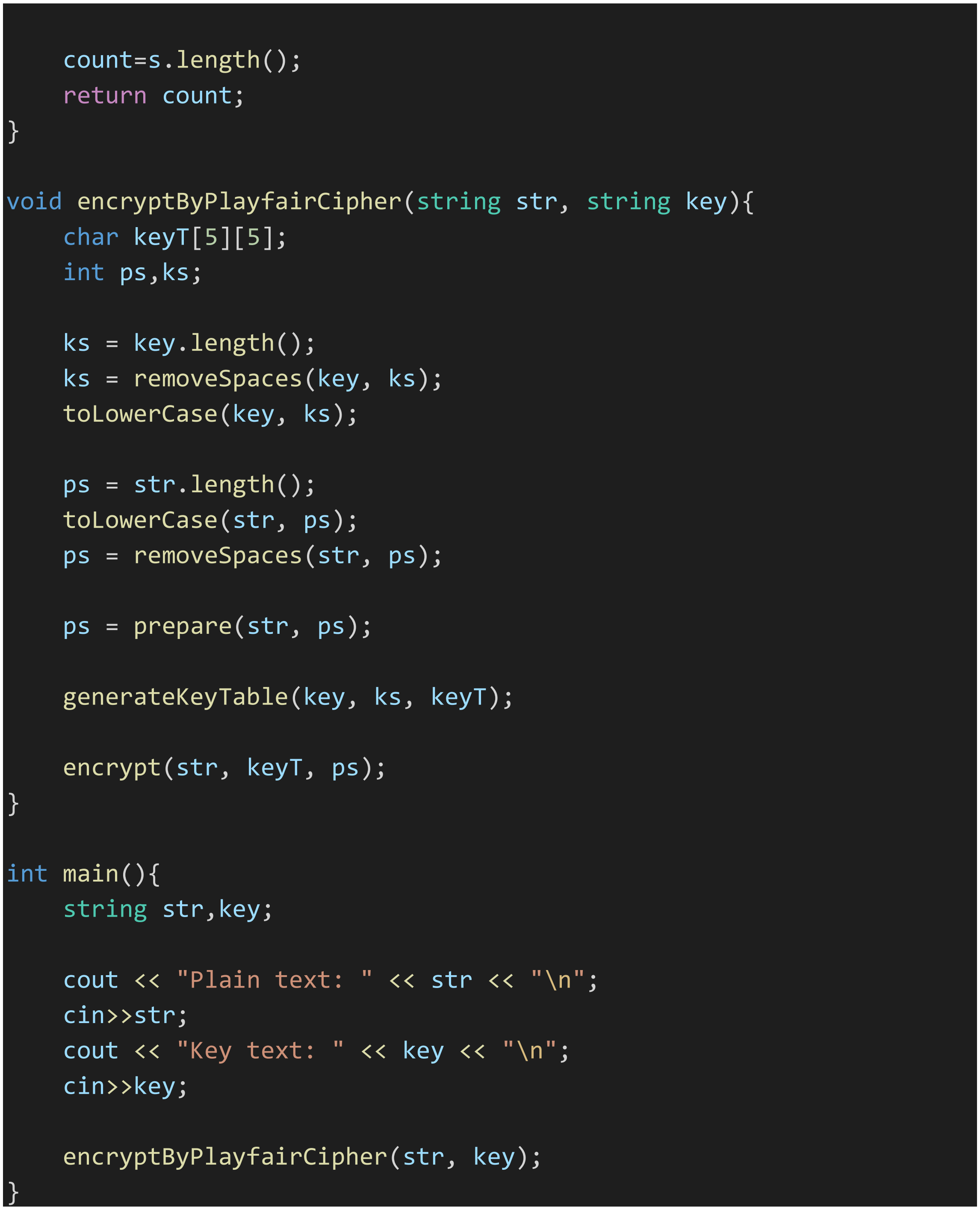
```
#include<bits/stdc++.h>
using namespace std;

int mod5(int a) { return (a % 5); }

// search letters in matrice
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;
            }
        }
    }
}

// gets the cipher text according to rules of playfair cipher
void encrypt(string str, char keyT[5][5], int ps){
    int i, a[4];

    for(i=0;i<5;i++){
        for(int j=0;j<5;j++){
            cout<<keyT[i][j]<<" ";
        }cout<<endl;
    }
```



```
    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]];
        }
    }
    cout << "Cipher text: " << str << "\n";

}

// generates 5x5 matrice
void generateKeyTable(string key, int ks, char keyT[5][5]){
    int i, j, k, flag = 0;

    int dicty[26] = { 0 };
    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++;
```

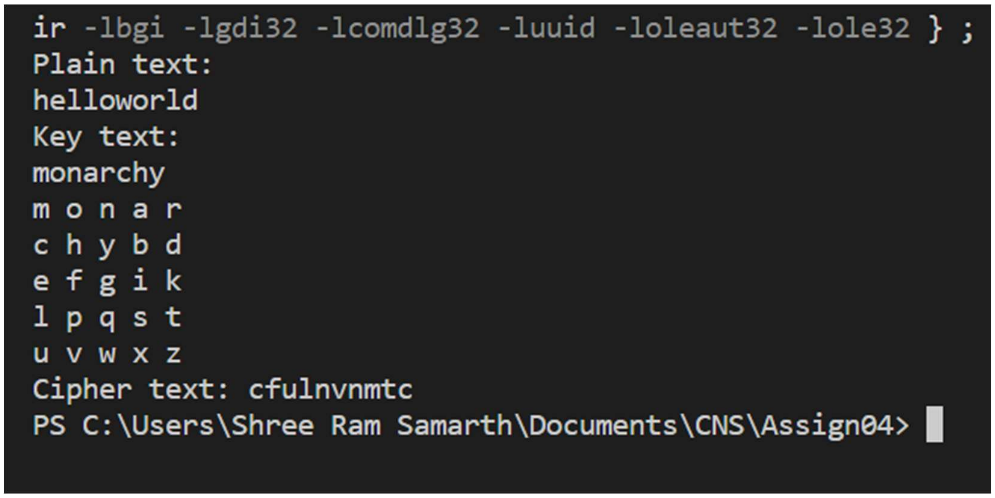


```
                j = 0;
            }
        }
    }

    for (k = 0; k < 26; k++) {
        if (dicty[k] == 0) {
            keyT[i][j] = (char)(k + 97);
            j++;
            if (j == 5) {
                i++;
                j = 0;
            }
        }
    }
}

// divides plain text into pairs
int prepare(string str, int ptrs){
    if (ptrs % 2 != 0) {
        str += 'z';
    }
    return str.length();
}

// converts to lowercase
void toLowerCase(string plain, int ps){
    int i;
    for (i = 0; i < ps; i++) {
        if (plain[i] > 64 && plain[i] < 91)
            plain[i] += 32;
    }
}

// removes spaces
int removeSpaces(string plain, int ps){
    int i, count = 0;
    string s;
    for (i = 0; i < ps; i++)
        if (plain[i] != ' ')
            s += plain[i];
    // plain[count] = '\0';
```



```
    count=s.length();
    return count;
}

void encryptByPlayfairCipher(string str, string key){
    char keyT[5][5];
    int ps,ks;

    ks = key.length();
    ks = removeSpaces(key, ks);
    toLowerCase(key, ks);

    ps = str.length();
    toLowerCase(str, ps);
    ps = removeSpaces(str, ps);

    ps = prepare(str, ps);

    generateKeyTable(key, ks, keyT);

    encrypt(str, keyT, ps);
}

int main(){
    string str,key;

    cout << "Plain text: " << str << "\n";
    cin>>str;
    cout << "Key text: " << key << "\n";
    cin>>key;

    encryptByPlayfairCipher(str, key);
}
```

Output:



```
ir -lbgi -lgdi32 -lcomdlg32 -luuid -loleaut32 -lole32 } ;
Plain text:
helloworld
Key text:
monarchy
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
Cipher text: cfulnvnmtc
PS C:\Users\Shree Ram Samarth\Documents\CNS\Assign04>
```