U18CO018

Shekhaliya Shubham

Lab Assignment 4

CNS

1. Implement a menu driven program for 5 X 5 Playfair Cipher with following functions.

   (a) Encrypt given plain text.

   (b) Decrypt given cipher text.

Code:

```
#include <bits/stdc++.h>

using namespace std;

char mat[5][5];
map<char, pair<int, int>> mapp;

void generateKeyMatrix(string key)
{

    replace(key.begin(), key.end(), 'j', 'i');
    set<char> set;

    int l = key.length();

    int i = 0, j = 0;

    for(int p = 0;p<l;p++) {

        if(set.find(key[p]) != set.end())
            continue;

        mat[i][j] = key[p];
        set.insert(key[p]);

        if(j==4) {
            i++;
        }
```

```
            j++;
            j %= 5;
        }


        for(char c = 'a';c<='z';c++) {
            if(set.find(c) != set.end())
                continue;

            if(set.find('j') != set.end() && c=='i')
                continue;

            if(c=='j')
                continue;

            mat[i][j] = c;

            if(j==4) {
                i++;
            }
            j++;
            j %= 5;
        }

        for(int i =0;i<5;i++) {
            for(int j = 0;j<5;j++) {
                cout<<mat[i][j]<<" ";
                mapp[mat[i][j]] = make_pair(i,j);
            }
            cout<<endl;
        }
    }

    string encrypt(string plainText) {

        string cipherText = "";

        vector<string> vec;

        int l = plainText.length();

        for(int i = 0;i<l;) {

            string ss = "";

            if(i+1 == l) {
                ss += plainText[i];
                if(isalpha(plainText[i]))
                    ss += "z";
```

```
            vec.emplace_back(ss);
            i++;
            continue;
        }


        char a = plainText[i];

        if(!isalpha(a)){
            ss += a;
            vec.emplace_back(ss);
            i++;
            continue;
        }

        char b = plainText[i+1];

        if(!isalpha(b)){
            ss += a;
            ss += "x";
            vec.emplace_back(ss);
            i+=2;
            ss = "";
            ss += b;
            vec.emplace_back(ss);
            continue;
        }

        if(a==b) {
            ss += a;
            ss += "x";
            vec.emplace_back(ss);
            i--;
        } else {
            ss += a;
            ss += b;
            vec.emplace_back(ss);
        }
        i+=2;
    }

    for(string ss : vec) {

        char a = ss[0];

        if(!isalpha(a)) {
            cipherText += ss;
            continue;
```

```
        }

        char b = ss[1];

        int i1 = mapp[a].first;
        int j1 = mapp[a].second;

        int i2 = mapp[b].first;
        int j2 = mapp[b].second;

        if(i1 == i2) { // same row

            j1 = (j1 + 1)%5;
            j2 = (j2 + 1)%5;

            cipherText += mat[i1][j1];
            cipherText += mat[i1][j2];
        } else if (j1 == j2) {      // same column

            i1 = (i1 + 1)%5;
            i2 = (i2 + 1)%5;

            cipherText += mat[i1][j1];
            cipherText += mat[i2][j1];
        } else {
            cipherText += mat[i1][j2];
            cipherText += mat[i2][j1];
        }
    }

    return cipherText;
}

string decrypt(string cipherText) {

    string plainText = "";

    int l = cipherText.length();

    for(int i = 0;i<l;) {

        if(!isalpha(cipherText[i])) {
            plainText += cipherText[i];
            i++;
            continue;
        }

        char a = cipherText[i];
```

```
        char b = cipherText[i+1];

        int i1 = mapp[a].first;
        int j1 = mapp[a].second;

        int i2 = mapp[b].first;
        int j2 = mapp[b].second;

        if(i1 == i2) {
            j1 = (j1 - 1 + 5)%5;
            j2 = (j2 - 1 + 5)%5;

            plainText += mat[i1][j1];
            plainText += mat[i1][j2];
        } else if(j1 == j2) {
            i1 = (i1 - 1 + 5)%5;
            i2 = (i2 - 1 + 5)%5;

            plainText += mat[i1][j1];
            plainText += mat[i2][j1];
        } else {
            plainText += mat[i1][j2];
            plainText += mat[i2][j1];
        }

        i+=2;
    }

    return plainText;
}

string readFrom(string filename)
{
    ifstream file;
    string input = "", result = "";
    file.open(filename);
    while (!file.eof())
    {
        getline(file, input);
        result += input + "\n";
    }
    file.close();
    return result.substr(0, result.length() - 1);
}

void writeTo(string filename, string message)
{
    ofstream file;
```

```
    file.open(filename);
    file << message;
    file.close();
}

int main() {

    string key = "default";
    generateKeyMatrix(key);

    int ch = 0;

    while(true) {

        cout<<"0. Key\n";
        cout<<"1. Encryption\n";
        cout<<"2. Decryption\n";

        cin>>ch;
        if(ch == 0) {
            cout<<"Enter the key: ";
            cin>>key;
            generateKeyMatrix(key);
        } else if(ch == 1) {

            string plainText = readFrom("input.txt");

            string cipherText = encrypt(plainText);
            writeTo("output1.txt", cipherText);

            cout<<"plain Text :: \n";
            cout<<plainText<<"\n\n";

            cout<<"cipher Text :: \n";
            cout<<cipherText<<"\n\n";

        } else if (ch == 2) {

            string cipherText = readFrom("output1.txt");

            string plainText = decrypt(cipherText);
            writeTo("output2.txt", plainText);

            cout<<"cipher Text :: \n";
            cout<<cipherText<<"\n\n";

            cout<<"plain Text :: \n";
            cout<<plainText<<"\n\n";
```

```
        } else {
            break;
        }
    }
    return 0;
}
```

Plain Text:

the playfair cipher or playfair square or wheatstone-playfair

cipher is a manual symmetric encryption!

technique and was the first literal digram substitution cipher.

Key: maytfh

Cipher Text:

mddz riytmylp bkobds ps riytmylp orvmsd ps ucbffrmrsn-riytmylp

bkobds np tv aygzti qftufhdxkb nsdqaqalsg!

fddbgkowdz filt vyrz mddz ansodt nkfdptrt blloya ozepalmxalsg bkobds.

Decrypted Text:

thex playfair cipher or playfair square or wheatstone-playfair

cipher is ax manual symxmetric encryption!

techniquex andx wasx thex firstx literalx digram substitution cipher.

```
0. Key
1. Encryption
2. Decryption
0
Enter the key: maytfh
m a y t f
h b c d e
g i k l n
o p q r s
u v w x z
0. Key
1. Encryption
2. Decryption
1
plain Text ::
the playfair cipher or playfair square or wheatstone-playfair
cipher is a manual symmetric encryption!
technique and was the first literal     digram substitution cipher.

cipher Text ::
mddz riytmylp bkobds ps riytmylp orvmsd ps ucbffrmrsn-riytmylp
bkobds np tv aygzti qftufhdxkb nsdqaqalsg!
fddbgkowdz filt vyrz mddz ansodt nkfdptrt     blloya ozepalmxalsg bkobds.

0. Key
1. Encryption
2. Decryption
2
cipher Text ::
mddz riytmylp bkobds ps riytmylp orvmsd ps ucbffrmrsn-riytmylp
bkobds np tv aygzti qftufhdxkb nsdqaqalsg!
fddbgkowdz filt vyrz mddz ansodt nkfdptrt     blloya ozepalmxalsg bkobds.

plain Text ::
thex playfair cipher or playfair square or wheatstone-playfair
cipher is ax manual symxmetric encryption!
techniquex andx wasx thex firstx literalx     digram substitution cipher.

0. Key
1. Encryption
2. Decryption
```