Program: 2 **Playfair Cipher**

Date:

**AIM**

**ALGORITHM**

**CODE**

#include <iostream>

#include <cmath>

#define BOGUT 'x'

using namespace std;

int in(string str, char c, int len) {

    for (int i=0; i<len; i++) {

        if ((char) str[i] == c) return 1;

    }

    return 0;

}

int mod(int num) {

    if (num < 0) {

        return 5 - (-1 \* num);

    }

    return num;

}

class PlayfairCipher {

    public:

    string keyword;

    char mat[5][5];

    PlayfairCipher(string key) {

        this->keyword = key;

        int l = key.length();

        int i, j, k;

        i = j = k = 0;

        while (k < l) {

            if (!in(this->keyword, (char) this->keyword[k], k)) {

                if (j == 5) {

                    i += 1;

                    j = 0;

                }

                this->mat[i][j] = (char) this->keyword[k];

                j++;

            }

            k++;

        }

        string alphas = "abcdefghiklmnopqrstuvwxyz";

        for (k=0; k<alphas.length(); k++) {

            if (!in(this->keyword, (char) alphas[k], l)) {

                if (j == 5) {

                    i += 1;

                    j = 0;

                }

                this->mat[i][j] = (char) alphas[k];

                j++;

            }

        }

        cout << "Matrix: " << endl;

        this->displayMatrix();

    }

    void encrypt(string plaintext) {

        int l = plaintext.length();

        char splitted[l][2];

        int k = 0;

        for (int i=0; i<l; i++) {

            splitted[k][0] = plaintext[i];

            if (i+1 >= l) {

                splitted[k][1] = BOGUT;

            } else if (plaintext[i] == plaintext[i+1]) {

                splitted[k][1] = BOGUT;

                k++;

                continue;

            } else {

                splitted[k][1] = plaintext[i+1];

            }

            i++;

            k++;

        }

        cout << "\nSplitted Plaintext: " << endl;

        for (int i=0; i<k; i++) {

            cout << splitted[i][0] << splitted[i][1] << " ";

        }

        cout << endl;

        cout << "\nCipher: " << endl;

        for (int i=0; i<k; i++) {

            int coord1[2];

            int coord2[2];

            findInMatrix(splitted[i][0], coord1);

            findInMatrix(splitted[i][1], coord2);

            if (coord1[0] == coord2[0]) {

                cout << this->mat[coord1[0]][(coord1[1] + 1) % 5]

                    << this->mat[coord2[0]][(coord2[1] + 1) % 5] << " ";

            } else if (coord1[1] == coord2[1]) {

                cout << this->mat[(coord1[0] + 1) % 5][coord1[1]]

                    << this->mat[(coord2[0] + 1) % 5][coord2[1]] << " ";

            } else {

                cout << this->mat[coord1[0]][coord2[1]]

                    << this->mat[coord2[0]][coord1[1]] << " ";

            }

        }

    }

    void decrypt(string cipher) {

        int l = cipher.length();

        if (l%2 != 0) {

            cout << "Cipher must be of even length." << endl;

            return;

        }

        cout << "\nPlaintext: " << endl;

        for (int i=0; i<l; i=i+2) {

            int coord1[2];

            int coord2[2];

            findInMatrix(cipher[i], coord1);

            findInMatrix(cipher[i+1], coord2);

            if (coord1[0] == coord2[0]) {

                cout << this->mat[coord1[0]][mod(coord1[1] - 1)]

                    << this->mat[coord2[0]][mod(coord2[1] - 1)];

            } else if (coord1[1] == coord2[1]) {

                cout << this->mat[mod(coord1[0] - 1)][coord1[1]]

                    << this->mat[mod(coord2[0] - 1)][coord2[1]];

            } else {

                cout << this->mat[coord1[0]][coord2[1]]

                    << this->mat[coord2[0]][coord1[1]];

            }

        }

    }

    void displayMatrix() {

        for (int i=0; i<5; i++) {

            for (int j=0; j<5; j++) {

                cout << this->mat[i][j] << " ";

            }

            cout << "\n";

        }

    }

    void findInMatrix(char c, int \*res) {

        res[0] = -1;

        res[1] = -1;

        for (int i=0; i<5; i++) {

            for (int j=0; j<5; j++) {

                if (this->mat[i][j] == c) {

                    res[0] = i;

                    res[1] = j;

                    return;

                }

            }

        }

    }

};

int main() {

    int choice;

    string key;

    cout << "\nEnter keyword: ";

    cin >> key;

    PlayfairCipher pc(key);

    while (1) {

        cout << "\n\n1. Encrypt" << endl;

        cout << "2. Decrypt" << endl;

        cout << "3. Exit" << endl;

        cout << "Enter Choice: ";

        cin >> choice;

        string text;

        if (choice == 1) {

            cout << "\nEnter plaintext: ";

            std::getline(std::cin >> std::ws, text);

            pc.encrypt(text);

        } else if (choice == 2) {

            cout << "\nEnter cipher: ";

            std::getline(std::cin >> std::ws, text);

            pc.decrypt(text);

        } else if (choice == 3) {

            cout << "Exiting.." << endl;

            break;

        } else {

            cout << "Invalid Choice" << endl;

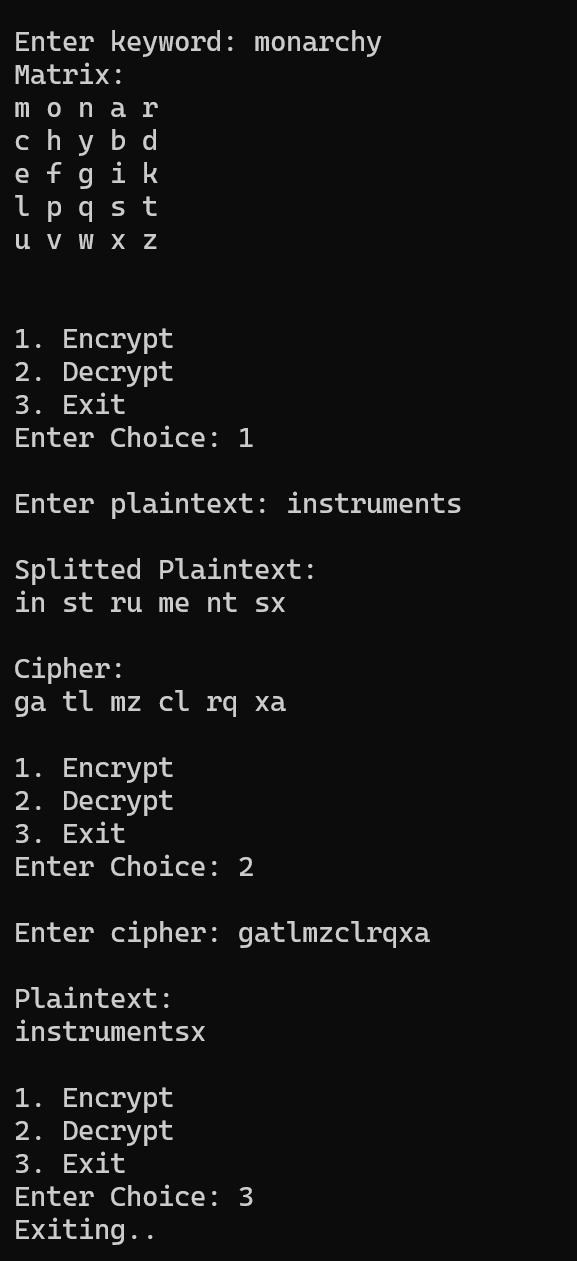
        }

    }

    return 0;

}

**OUTPUT**

****

**RESULT**

Thus, the program to implement encryption and decryption using Playfair cipher is successfully completed.