**Cryptography and Network Security Lab**

**Name: Rushikesh R. Ware**

**PRN: 2020BTECS00012**

**Batch: B2**

**COLUMNAR TRANSPOSITION CIPHER & RAIL FENCE ALGORITHM**

**Aim:**

To encrypt given plain text using columnar cipher.

**Theory:**

Columnar cipher is a type of transposition cipher. Columnar cipher involves writing the plaintext in rows and columns and then reading columns one by one.

**Code:**

#include <bits/stdc++.h>

using namespace std;

#define ll long long

int main()

{

    string plainText, key;

    cout << "\n Choose an option:\n";

    cout << "  1. Encryption\n";

    cout << "  2. Decryption\n";

    int choice;

    cin >> choice;

    cin.ignore();

    if (choice == 1)

    {

        // Encryption

        cout << "\n Enter plain text : ";

        getline(cin, plainText);

        cout << "\n Enter key : ";

        getline(cin, key);

        // Removing spaces and converting to lowercase from plaintext

        string temp = "";

        for (int i = 0; i < plainText.size(); i++)

        {

            if (plainText[i] != ' ')

                temp += tolower(plainText[i]);

        }

        plainText = temp;

        // Removing spaces and converting to lowercase from key

        string temp2 = "";

        for (int i = 0; i < key.size(); i++)

        {

            if (key[i] != ' ')

                temp2 += tolower(key[i]);

        }

        key = temp2;

        // Encryption

        map<char, vector<char>> mp;

        int keyCounter = 0;

        for (int i = 0; i < plainText.size(); i++)

        {

            mp[key[keyCounter++]].push\_back(plainText[i]);

            if (keyCounter == key.size())

                keyCounter = 0;

        }

        string cipherText;

        for (auto it : mp)

        {

            for (int i = 0; i < it.second.size(); i++)

            {

                cipherText += it.second[i];

            }

        }

        cout << "\n Cipher text is : " << cipherText << endl;

    }

    else if (choice == 2)

    {

        // Decryption

        cout << "\n Enter cipher text : ";

        getline(cin, plainText);

        cout << "\n Enter key : ";

        getline(cin, key);

        // Removing spaces and converting to lowercase from key

        string temp2 = "";

        for (int i = 0; i < key.size(); i++)

        {

            if (key[i] != ' ')

                temp2 += tolower(key[i]);

        }

        key = temp2;

        // Decryption

        map<int, int> dmp;

        int common = plainText.size() / key.size();

        int extra = plainText.size() % key.size();

        for (int i = 0; i < key.size(); i++)

        {

            if (i < extra)

                dmp[i] = common + 1;

            else

                dmp[i] = common;

        }

        map<int, vector<char>> dmp2;

        int start = 0;

        string sortedKey = key;

        sort(sortedKey.begin(), sortedKey.end());

        for (int i = 0; i < sortedKey.size(); i++)

        {

            for (int j = 0; j < key.size(); j++)

            {

                if (sortedKey[i] == key[j])

                {

                    for (int k = 0; k < dmp[j]; k++)

                    {

                        dmp2[key[j]].push\_back(plainText[start++]);

                    }

                }

            }

        }

        string afterDecryption;

        vector<int> counters(key.size(), 0);

        int i = 0;

        while (afterDecryption.size() < plainText.size())

        {

            for (int i = 0; i < key.size(); i++)

            {

                if (counters[i] < dmp[i])

                    afterDecryption += dmp2[key[i]][counters[i]++];

            }

        }

        cout << "\n\n Text after decryption is : " << afterDecryption << endl;

    }

    else

    {

        cout << "\n Invalid choice" << endl;

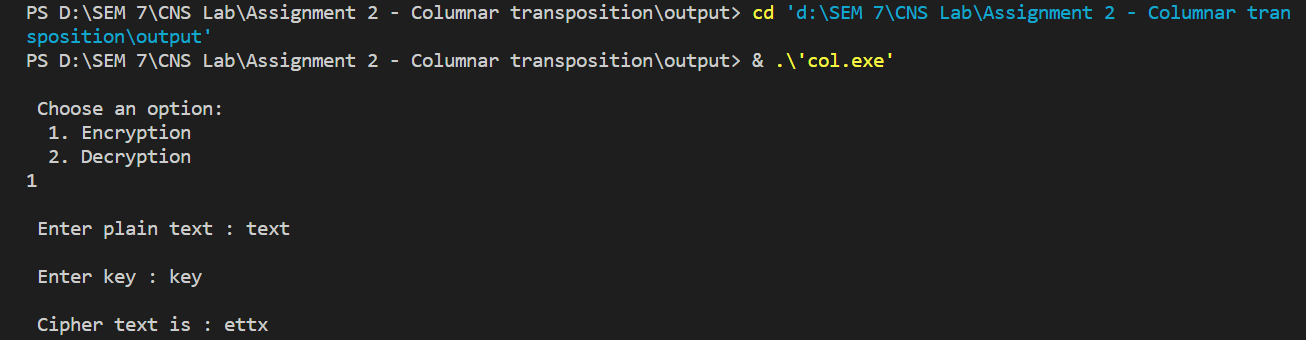
    }

    return 0;

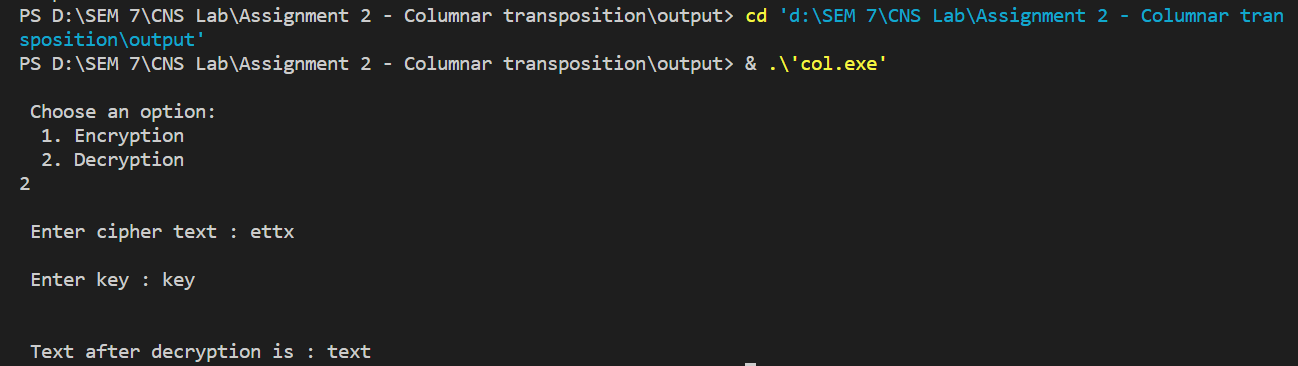
}

**Output:**

**Encryption:**



**Decryption:**



**Aim:**

To encrypt given plain text using rail fence cipher.

**Theory:**

Railfence cipher is a transposition type of cipher, it is also known as zigzag cipher. And its encryption algorithm is similar to fence of horizontal rails, so it is known as rail fence cipher.

**Code:**

#include <iostream>

#include <bits/stdc++.h>

using namespace std;

int main()

{

    string s; cout << "\nEnter plain text" << endl; getline(cin, s);

    string x; for (int i = 0; i < s.length(); i++)

    if (s[i] != ' ') x += s[i];

    s = x;

    int k;

    cout << "\nEnter key" << endl; cin >> k;

    cout << "\nPlain text is: " << s << endl; cout << "\nKey is: " << k << endl; int n = s.length();

    vector<vector<char>> mat(k); int row = 0; int flg = 1;

    for (int i = 0; i < s.length(); i++)

    {

    mat[row].push\_back(s[i]); row += flg; if (row == (k - 1))

    { flg = -1;

    } if (row == 0) flg = 1;

    }

    string cip = ""; for (int i = 0; i < k; i++) { for (int j = 0; j < mat[i].size(); j++) cip += mat[i][j];

    }

    s = cip; transform(cip.begin(), cip.end(), cip.begin(), ::toupper); cout << "\nCipher text is: " << cip;

    int tp = 1;

    vector<vector<int>> matd(k); row = 0; flg = 1;

    for (int i = 1; i <= n; i++)

    {

    matd[row].push\_back(i); row += flg; if (row == (k - 1))

    { flg = -1;

    } if (row == 0) flg = 1;

    }

    vector<int> dd;

    for (int i = 0; i < k; i++)

    { for (int j = 0; j < mat[i].size(); j++) dd.push\_back(matd[i][j]);

    }

    cout << endl; map<int, char> m;

    for (int i = 0; i < n; i++) m[dd[i]] = s[i];

    string plain = ""; for (int i = 1; i <= n; i++) plain += m[i];

    cout << "\n\nPlain text after decription is: " << plain;

    return 0;

}

**Output:**

