**Assignment No 3**

**Name-Aniket Tatoba Mane**

**PRN-2020BTECS00020**

**Batch-B7**

**Subject - CNS LAB**

**Aim**: To implement Columnar cipher

**Theory**: The Columnar Cipher is a transposition cipher where the plaintext is written out in rows and then read out again column by column in sorted order of characters in key. The key determines the number of columns.

Code:

#include <bits/stdc++.h>

using namespace std;

string alpha\_lower(string text)

{

    for (char c : text)

    {

        if (isalnum(c))

        {

            c = tolower(c);

        }

    }

    return text;

}

string encrypt(string text, string key)

{

    map<char, vector<char>> mp;

    int cnt = 0;

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

    {

        if (cnt == key.size())

            cnt = 0;

        mp[key[cnt++]].push\_back(text[i]);

    }

    string encrypted;

    for (auto i : mp)

    {

        for (auto j : i.second)

        {

            encrypted += j;

        }

    }

    return encrypted;

}

string decrypt(string cipher, string key)

{

    map<int, int> map1;

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

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

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

    {

        if (i < extra)

            map1[i] = common + 1;

        else

            map1[i] = common;

    }

    map<int, vector<char>> map2;

    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 < map1[j]; k++)

                {

                    map2[key[j]].push\_back(cipher[start++]);

                }

            }

        }

    }

    string plain;

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

    while (plain.size() < cipher.size())

    {

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

        {

            if (counters[i] < map1[i])

                plain += map2[key[i]][counters[i]++];

        }

    }

    return plain;

}

int main()

{

    int choice;

    cout << "Enter choice: ";

    cout << endl

         << "1. Encrypt | 2. Decrypt" << endl;

    cin >> choice;

    cin.get();

    if (choice == 1)

    {

        string text, key;

        cout << "\nEnter text: ";

        getline(cin, text);

        text = alpha\_lower(text);

        cout << "\nEnter key: ";

        getline(cin, key);

        alpha\_lower(key);

        string cipher = encrypt(text, key);

        cout << "\nEncrypted text is : " << cipher << endl;

    }

    else if (choice == 2)

    {

        string cipher, key;

        cout << "\nEnter cipher text: ";

        getline(cin, cipher);

        cipher = alpha\_lower(cipher);

        cout << "\nEnter key: ";

        getline(cin, key);

        alpha\_lower(key);

        string text = decrypt(cipher, key);

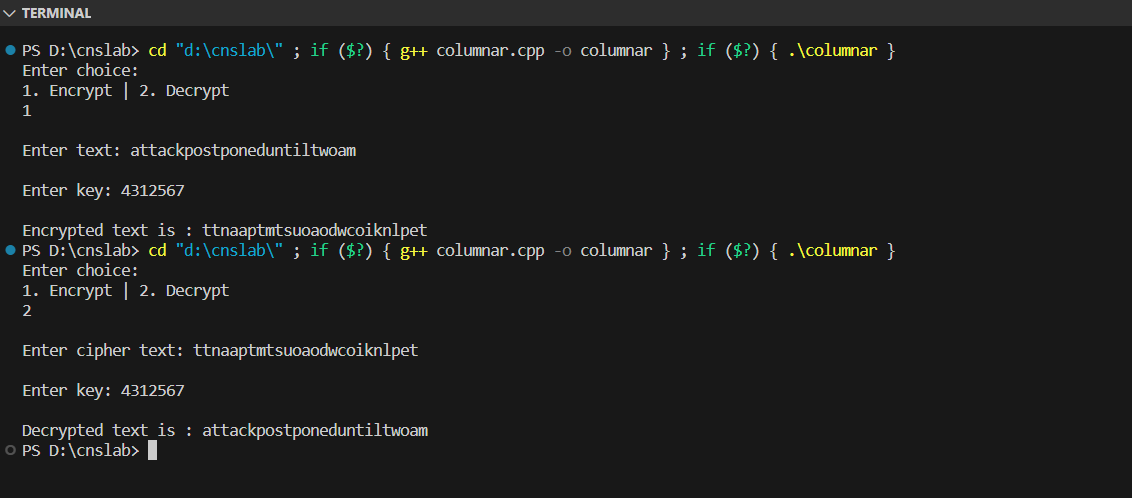
        cout << "\nDecrypted text is : " << text << endl;

    }

    return 0;

}

Output:



Advantages:

Key-Dependent Security: The Columnar Cipher's security is heavily dependent on the choice of the key, which determines the arrangement of letters in the grid. If a strong and random key is used, it can provide a reasonable level of security.

Simplicity: The Columnar Cipher is relatively simple to understand and implement. It does not involve complex mathematical operations or require large key sizes, making it accessible for educational purposes and simple cryptographic tasks.