**Final Year B.Tech. (CSE) – II [ 2021-22 ]**

**Cryptograpy and Network Security Lab**

**PRN: 2019BTECS00015**

**Full name: Shraddha Sanjay Kharat**

**Batch: B1**

**Assignment no -5**

**Date: 23-08-2022**

**Que )** **Rail Fence & Columnar Transposition Cipher**

**Aim: To Demonstrate Rail Fence & Columnar Transposition Cipher**

**Theory:**

The rail fence cipher (also called a zigzag cipher) is a form of transposition cipher. It derives its name from the way in which it is encoded. The Columnar Transposition Cipher is a form of transposition cipher just like Rail Fence Cipher. Columnar Transposition involves writing the plaintext out in rows, and then reading the ciphertext off in columns one by one.

**Procedure:**

Rail Fence Encryption

1. In the rail fence cipher, the plain-text is written downwards and diagonally on successive rails of an imaginary fence
2. When we reach the bottom rail, we traverse upwards moving diagonally, after reaching the top rail, the direction is changed again. Thus the alphabets of the message are written in a zig-zag manner
3. After each alphabet has been written, the individual rows are combined to obtain the cipher-text

Rail Fence Decryption

1. The number of columns in rail fence cipher remains equal to the length of plain-text message. And the key corresponds to the number of rails
2. Rail matrix can be constructed accordingly. Once we’ve got the matrix we can figure-out the spots where texts should be placed (using the same way of moving diagonally up and down alternatively)
3. Then, we fill the cipher-text row wise. After filling it, we traverse the matrix in zig-zag manner to obtain the original text

Columnar Transposition Encryption

1. The message is written out in rows of a fixed length, and then read out again column by column
2. Width of the rows and the permutation of the columns are usually defined by a keyword
3. Any spare spaces are filled with nulls or left blank or placed by a character
4. Finally, the message is read off in columns, in the order specified by the keyword

Columnar Transposition Decryption

1. To decipher it, the recipient has to work out the column lengths by dividing the message length by the key length
2. Then, write the message out in columns again, then re-order the columns by reforming the key word

**Code Snapshots:**

1. **Rail Fence Cipher**

#include <bits/stdc++.h>

using namespace std;

string RailFenceEncrypt(string text, int key)

{

    string result;

    char rail[key][text.length()];

    for (int i = 0; i < key; i++)

    {

        for (int j = 0; j < text.length(); j++)

        {

            rail[i][j] = '\n';

        }

    }

    bool dir\_down;

    int row = 0, col = 0;

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

    {

        if (row == 0)

            dir\_down = true;

        else if (row == key - 1)

            dir\_down = false;

        rail[row][col++] = text[i];

        if (dir\_down)

            row++;

        else

            row--;

    }

    for (int i = 0; i < key; i++)

    {

        for (int j = 0; j < text.length(); j++)

        {

            if (rail[i][j] != '\n')

                result.push\_back(rail[i][j]);

        }

    }

    return result;

}

string RailFenceDecrypt(string ciphertext, int key)

{

    string result;

    char rail[key][ciphertext.length()];

    for (int i = 0; i < key; i++)

    {

        for (int j = 0; j < ciphertext.length(); j++)

        {

            rail[i][j] = '\n';

        }

    }

    bool dir\_down;

    int row = 0, col = 0;

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

    {

        if (row == 0)

            dir\_down = true;

        else if (row == key - 1)

            dir\_down = false;

        rail[row][col++] = '\*';

        if (dir\_down)

            row++;

        else

            row--;

    }

    int index = 0;

    for (int i = 0; i < key; i++)

    {

        for (int j = 0; j < ciphertext.length(); j++)

        {

            if (rail[i][j] == '\*' && index < ciphertext.length())

                rail[i][j] = ciphertext[index++];

        }

    }

    row = 0, col = 0;

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

    {

        if (row == 0)

            dir\_down = true;

        else if (row == key - 1)

            dir\_down = false;

        if (rail[row][col] != '\*')

            result.push\_back(rail[row][col++]);

        if (dir\_down)

            row++;

        else

            row--;

    }

    return result;

}

int main()

{

    int option, key;

    string text, ciphertext;

    cout << "Enter option:\n1)Console\n2)File\n";

    cin >> option;

    cout << "Enter key: ";

    cin >> key;

    cin.ignore();

    switch (option)

    {

    case 1:

        cout << "Enter text: ";

        break;

    case 2:

        freopen("input.txt", "r", stdin);

        freopen("output.txt", "w", stdout);

        break;

    }

    getline(cin, text);

    ciphertext = RailFenceEncrypt(text, key);

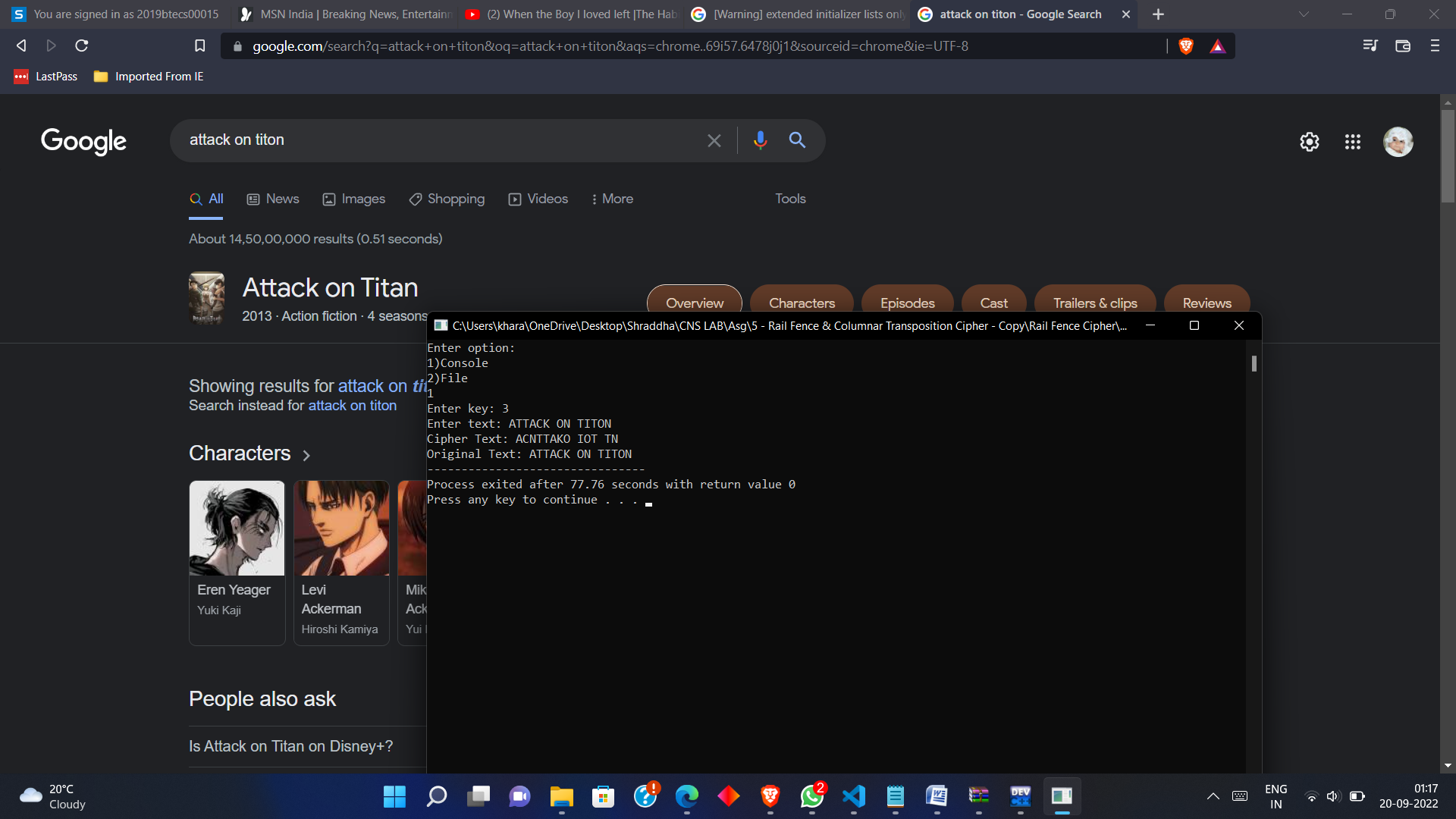
    cout << "Cipher Text: " << ciphertext << endl;

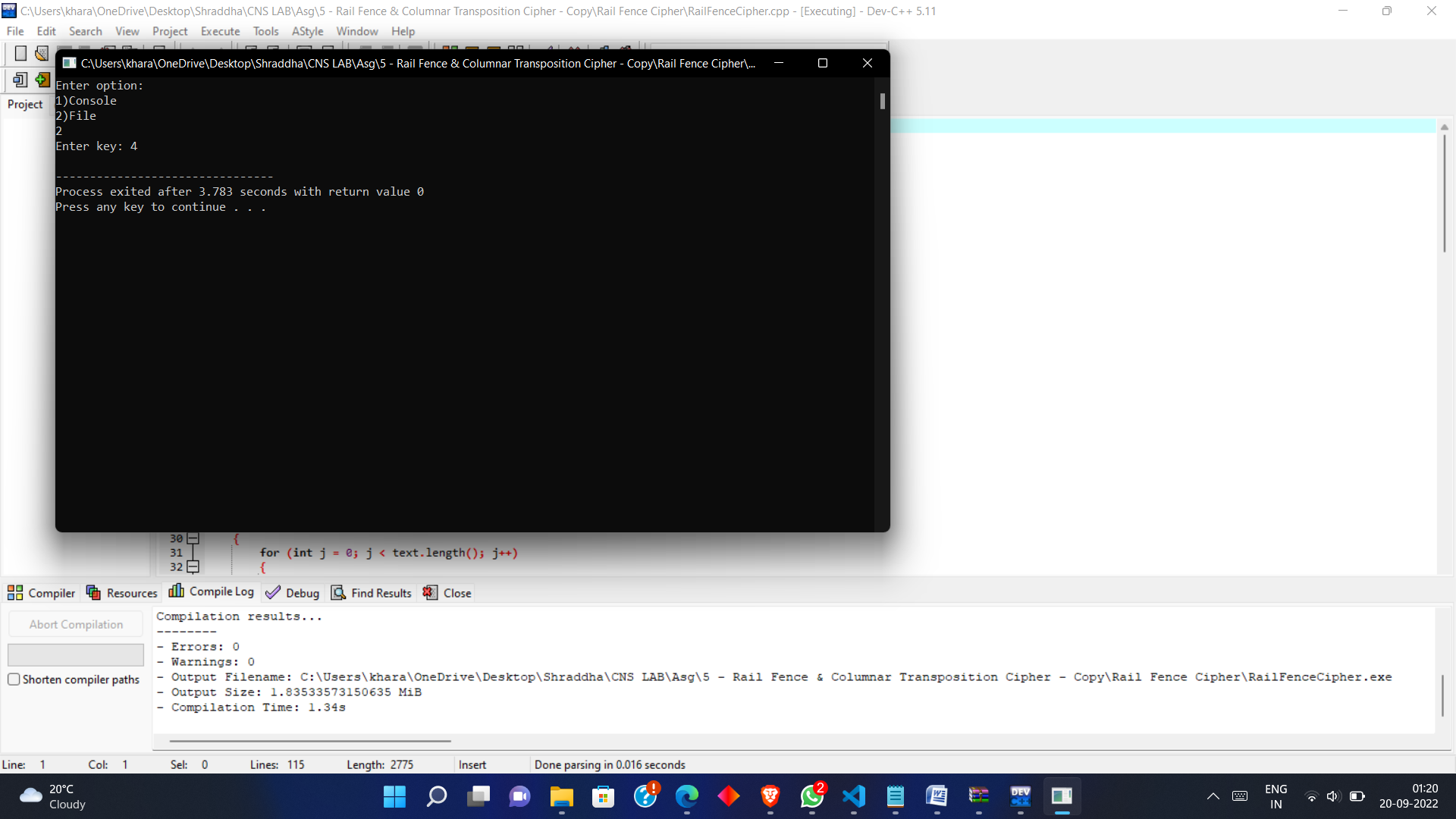
    cout << "Original Text: " << RailFenceDecrypt(ciphertext, key);

    return 0;

}

**Output:**





Cipher Text: D ALEDTESALFNH TWEE

Original Text: DEFEND THE EAST WALL

**Code Snapshot :**

1. **Columnar Transposition Cipher**

#include <bits/stdc++.h>

using namespace std;

void SetPermutationOrder(string key, map<int, int> &keyMap)

{

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

    {

        keyMap[key[i]] = i;

    }

}

string ColumnarTranspositionEncrypt(string text, string key)

{

    string result = "";

    map<int, int> keyMap;

    int row, col, x = 0;

    col = key.length();

    row = text.length() / col;

    if (text.length() % col)

        row += 1;

    char matrix[row][col];

    for (int i = 0; i < row; i++)

    {

        for (int j = 0; j < col; j++)

        {

            if (x < text.size())

                matrix[i][j] = text[x];

            else

                matrix[i][j] = '\_';

            x++;

        }

    }

    SetPermutationOrder(key, keyMap);

    for (auto itr : keyMap)

    {

        for (int i = 0; i < row; i++)

        {

            result += matrix[i][itr.second];

        }

    }

    return result;

}

string ColumnarTranspositionDecrypt(string ciphertext, string key)

{

    string result = "";

    map<int, int> keyMap;

    int col = key.length();

    int row = ciphertext.length() / col;

    char cipherMat[row][col];

    int x = 0;

    for (int i = 0; i < col; i++)

    {

        for (int j = 0; j < row; j++)

        {

            cipherMat[j][i] = ciphertext[x++];

        }

    }

    SetPermutationOrder(key, keyMap);

    x = 0;

    for (auto itr = keyMap.begin(); itr != keyMap.end(); itr++)

    {

        itr->second = x++;

    }

    for (int i = 0; i < row; i++)

    {

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

        {

            char c = cipherMat[i][keyMap[key[j]]];

            if (c != '\_')

                result += c;

        }

    }

    return result;

}

int main()

{

    int option;

    string key, text, ciphertext;

    cout << "Enter option:\n1)Console\n2)File\n";

    cin >> option;

    cout << "Enter key: ";

    cin >> key;

    cin.ignore();

    switch (option)

    {

    case 1:

        cout << "Enter text: ";

        break;

    case 2:

        freopen("input.txt", "r", stdin);

        freopen("output.txt", "w", stdout);

        break;

    }

    getline(cin, text);

    ciphertext = ColumnarTranspositionEncrypt(text, key);

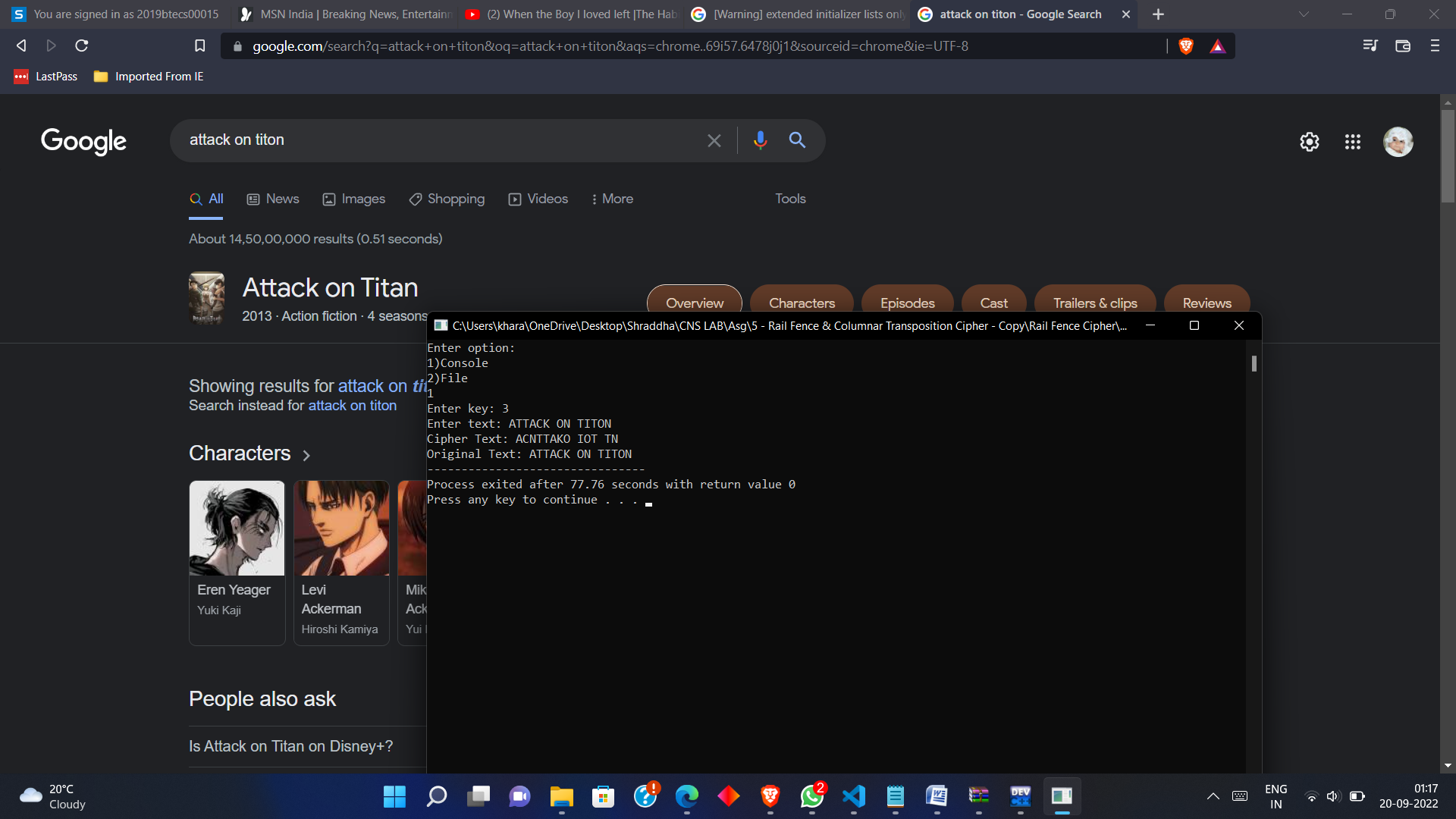
    cout << "Cipher Text: " << ciphertext << endl;

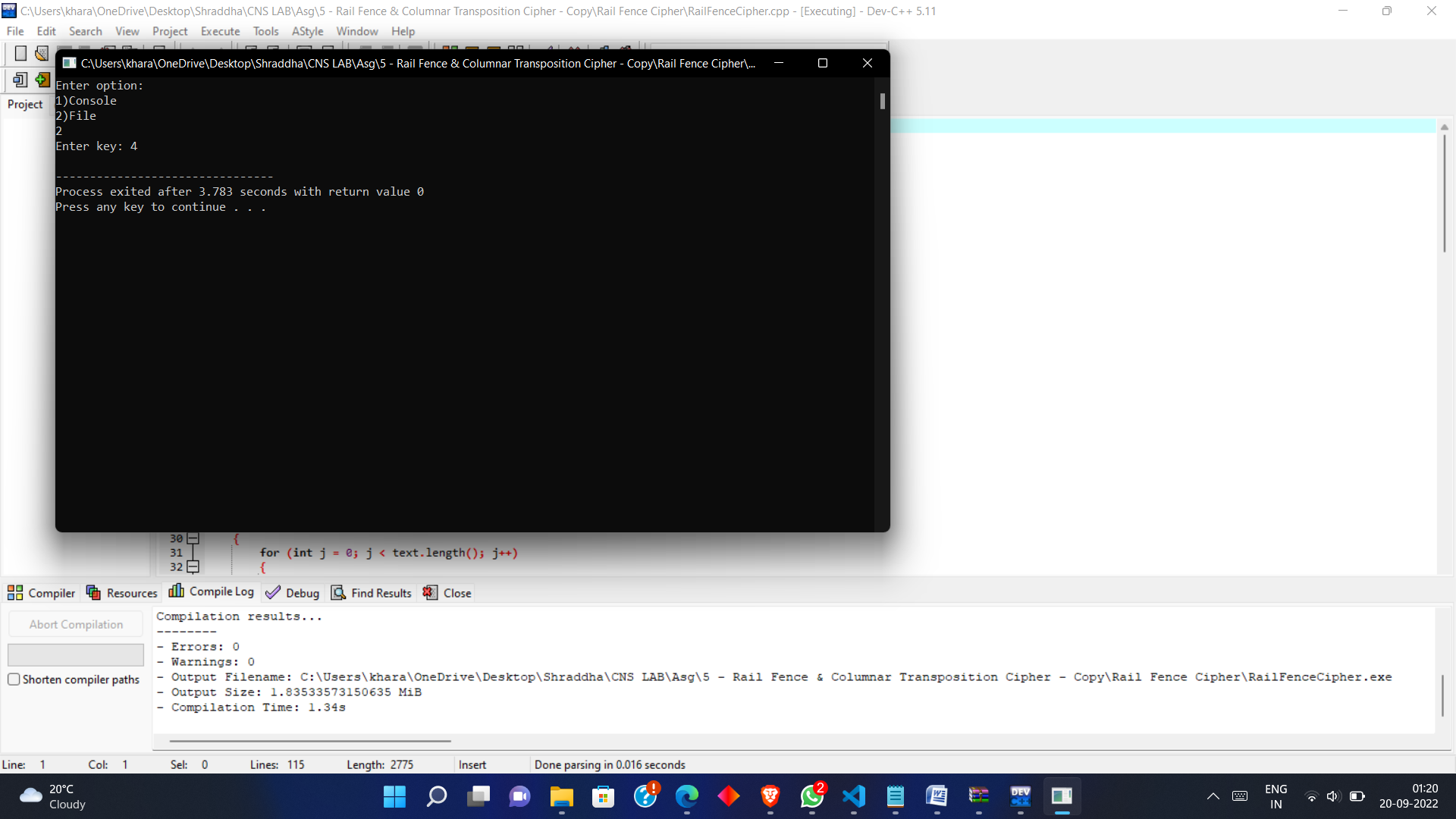
    cout << "Original Text: " << ColumnarTranspositionDecrypt(ciphertext, key);

    return 0;

}

**Output:**

****

****

Cipher Text: E  KEFGSGSREKOE\_

Original Text: GEEKS FOR GEEKS

* **Conclusion** :

**Cryptanalysts observed a significant improvement in crypto security when transposition technique is performed. They also noted that re-encrypting the cipher text using same transposition cipher creates better security. A double columnar transposition was used by the U.S. Army in World War I, and it is just a columnar transposition followed by another columnar transposition.**