Cryptography & Network Security

PRN - 2019BTECS00026
Name - Niraja Vasudev Kulkarni
Batch - B1

Assignment - 5

Transposition Ciphers - Railfence & Columnar

- **Objective -**

  To implement Railfence cipher & Columnar cipher in C

- **Theory -**

In a transposition cipher, the order of the alphabets is re-arranged to obtain the cipher-text. In the rail fence cipher, the plain-text is written downwards and diagonally on successive rails of an imaginary fence. 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. After each alphabet has been written, the individual rows are combined to obtain the cipher-text.

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.

- **Code Snapshots -**

Railefence Cipher :

```
#include <bits/stdc++.h>
using namespace std;
```

```
string encrypt(string plainText, int depth)
{
    char matrix[depth][(plainText.length())];
    for (int i=0; i < depth; i++)
        for (int j = 0; j < plainText.length(); j++)
            matrix[i][j] = '*';
    int d = -1;
    int r = 0, c = 0;
```

```
    for (int i=0; i < plainText.length(); i++)
    {
        if (r == 0 || r == depth-1)
            d = -1*d;
        matrix[r][c++] = plainText[i];
        if(d==1) r++; //d==1 means we should go down in the matrix
        else r--;
    }
    string cipherText;
    for (int i=0; i < depth; i++)
        for (int j=0; j < plainText.length(); j++)
```

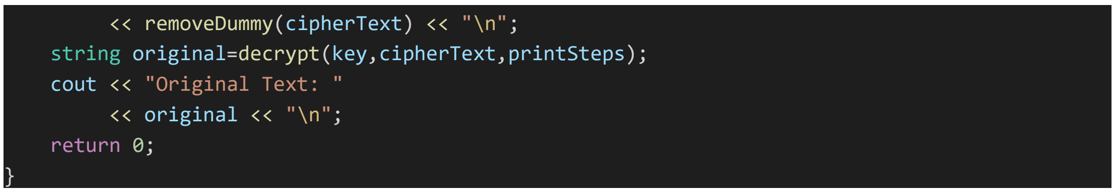
```
            if (matrix[i][j]!='*')
                cipherText.push_back(matrix[i][j]);
```

```
    return cipherText;
}
int main()
{
    int option;
    cout << "How do you want to give input?:\n1) Through terminal\n2) Through File\n";
    cin >> option;
    string plainText;
    int depth;
    cout << "Enter Depth for RailFence cipher: ";
    cin >> depth;
    switch (option)
    {
    case 1:
        cout << "Enter Text: ";
        break;
    case 2:
        freopen("input.txt", "r", stdin);
        freopen("output.txt", "w", stdout);
        break;
    default:
        break;
    }
    cin >> plainText;
    cout << "Cipher Text: "
         << encrypt(plainText, depth) << "\n";
    return 0;
}
```

Columnar Cipher -

```
#include <bits/stdc++.h>
using namespace std;

string encrypt(string key, string plainText,bool printSteps) {
vector <int> pos;
    for(int i=1;i<=key.length();i++)
    {
        int j;
        for(j=0;j<key.length();j++)
        {
            if(key[j]==char(i+48))
            break;
        }
        pos.push_back(j);
    }
    string cipherText="";
    int k=0,j;
    int rows=(int)ceil(floor(plainText.length())/floor(key.length()));
    int cols=key.length();
    vector<vector<char>> mat( rows , vector<char> (cols, 0));
    for(int i=0;i<rows;i++)
    {
        for( j=0;j<cols;j++)
        {
            mat[i][j]='*';
        }
    }
    for(int i=0;i<rows;i++)
    {
        for( j=0;j<cols;j++)
        {
            if(plainText[k]=='\0')
              break;
            mat[i][j]=plainText[k++];
        }
        if(j!=key.length())
        break;
    }
    if(printSteps)
    {
        cout<<"\nMatrix for columnar encryption:\n";
     for(int i=0;i<rows;i++)
    {
        for(int j=0;j<key.length();j++)
        {
            cout<<mat[i][j]<<" ";
        }
        cout<<"\n";
    }
    }
    int cnt=0;
    for(int loop=0;loop<cols;loop++)
    {
        for(int j=0;j<rows;j++)
        {
                cipherText.push_back(mat[j][pos[loop]]);
        }
    }
  return cipherText;

}
string decrypt(string key, string cipherText,bool printSteps)
 {

vector <int> pos;
    for(int i=1;i<=key.length();i++)
    {
        int j;
        for(j=0;j<key.length();j++)
        {
            if(key[j]==char(i+48))
            break;
        }
        pos.push_back(j);
    }
    string plainText="";
    int j,cnt=0;
    int rows=(int)ceil(floor(cipherText.length())/floor(key.length()));
    int cols=key.length();
    vector<vector<char>> mat( rows , vector<char> (cols, 0));
    for(int i=0;i<cols;i++)
    {
        for( j=0;j<rows;j++)
        {
            mat[j][pos[i]]=cipherText[cnt++];
        }
```

```cpp
    }
    if(printSteps){
cout<<"\nMatrix for columnar decryption:\n";
     for(int i=0;i<rows;i++)
    {
        for(int j=0;j<cols;j++)
        {
            cout<<mat[i][j]<<" ";
        }
        cout<<"\n";
    }
    }


    for(int i=0;i<rows;i++)
    {
        for(int j=0;j<cols;j++)
        {
          if(mat[i][j]!='*')
          plainText.push_back(mat[i][j]);
        }
    }
  return plainText;

}
string removeDummy(string cipherText)
{
    string ans;
    for(int i=0;i<cipherText.size();i++)
    {
        if(cipherText[i]!='*')
        ans.push_back(cipherText[i]);
    }
    return ans;
}
int main()
{
  int option;
    cout << "How do you want to give input?:\n1) Through terminal\n2) Through File\n";
    cin >> option;
    string plainText;
    string key;
    bool printSteps=false;
    cout << "Enter key: ";
    cin >> key;
    switch (option)
    {
    case 1:
        cout << "Enter Text: ";
        break;
    case 2:
        freopen("input.txt", "r", stdin);
        freopen("output.txt", "w", stdout);
        break;
    default:
        break;
    }
    if(option==1) printSteps=true;
    cin >> plainText;
    string cipherText=encrypt(key,plainText,printSteps);
    cout << "Cipher Text: "
```

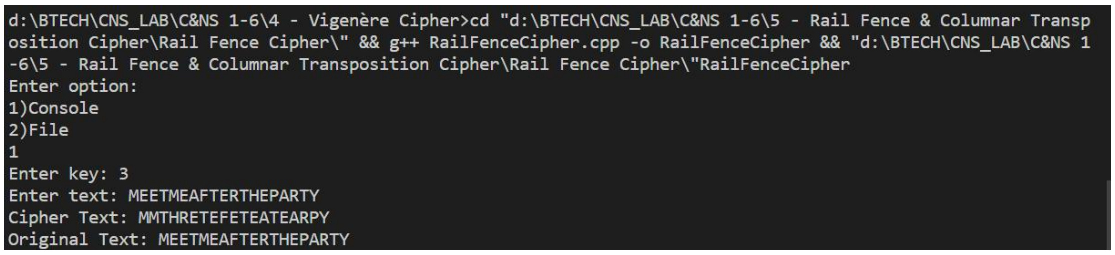


```
        << removeDummy(cipherText) << "\n";
    string original=decrypt(key,cipherText,printSteps);
    cout << "Original Text: "
        << original << "\n";
    return 0;
}
```

- **Outputs -**

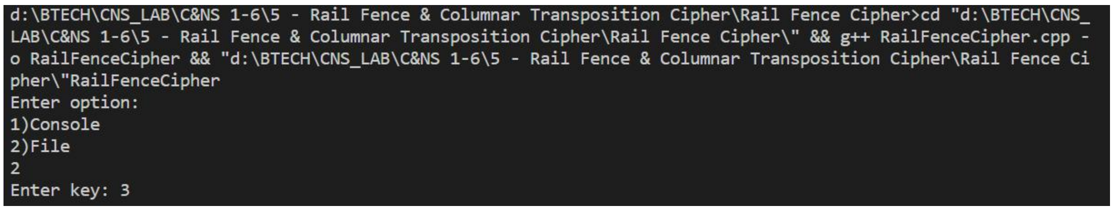
1. Railfence Cipher :

- **Sample Output 1 -**



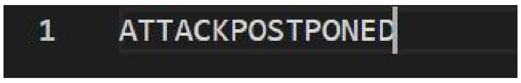
```
d:\BTECH\CNS_LAB\C&NS 1-6\4 - Vigenère Cipher>cd "d:\BTECH\CNS_LAB\C&NS 1-6\5 - Rail Fence & Columnar Transp
osition Cipher\Rail Fence Cipher\" && g++ RailFenceCipher.cpp -o RailFenceCipher && "d:\BTECH\CNS_LAB\C&NS 1
-6\5 - Rail Fence & Columnar Transposition Cipher\Rail Fence Cipher\"RailFenceCipher
Enter option:
1)Console
2)File
1
Enter key: 3
Enter text: MEETMEAFTERTHEPARTY
Cipher Text: MMTHRETEFETEATEARPY
Original Text: MEETMEAFTERTHEPARTY
```

- **Sample Output 2 -**



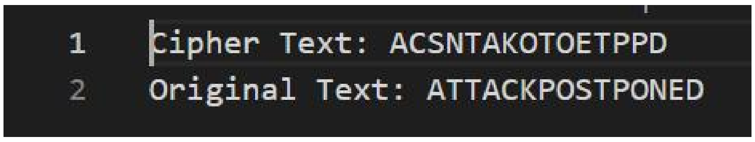
```
d:\BTECH\CNS_LAB\C&NS 1-6\5 - Rail Fence & Columnar Transposition Cipher\Rail Fence Cipher>cd "d:\BTECH\CNS_
LAB\C&NS 1-6\5 - Rail Fence & Columnar Transposition Cipher\Rail Fence Cipher\" && g++ RailFenceCipher.cpp -
o RailFenceCipher && "d:\BTECH\CNS_LAB\C&NS 1-6\5 - Rail Fence & Columnar Transposition Cipher\Rail Fence Ci
pher\"RailFenceCipher
Enter option:
1)Console
2)File
2
Enter key: 3
```

Input file -



```
ATTACKPOSTPONED
```

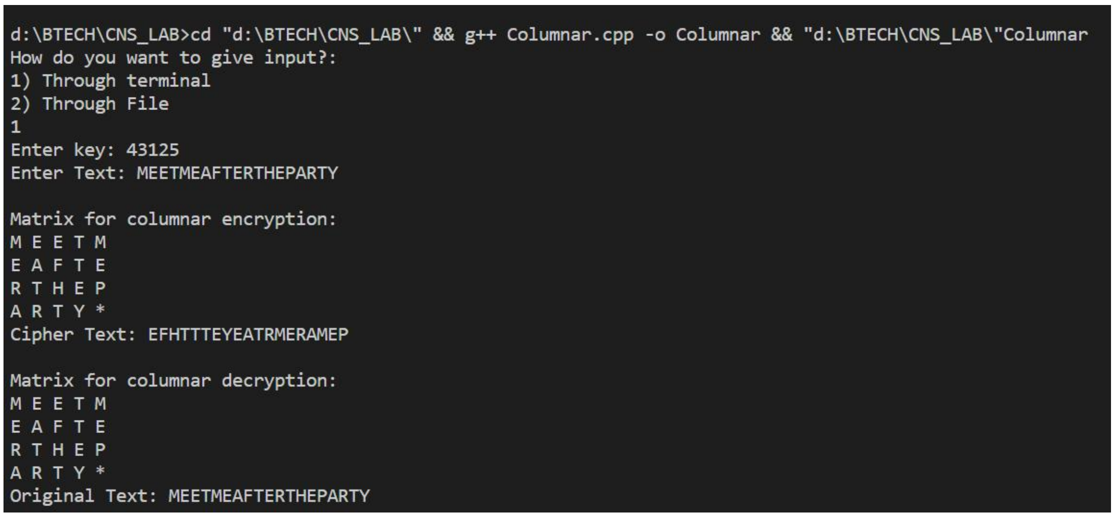
Output file -



```
Cipher Text: ACSNTAKOTOETPPD
Original Text: ATTACKPOSTPONED
```

2. Columar Cipher -

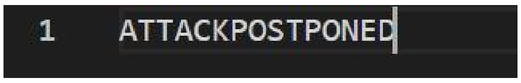
- Sample Output 1 -



```
d:\BTECH\CNS_LAB>cd "d:\BTECH\CNS_LAB\" && g++ Columnar.cpp -o Columnar && "d:\BTECH\CNS_LAB\"Columnar
How do you want to give input?:
1) Through terminal
2) Through File
1
Enter key: 43125
Enter Text: MEETMEAFTERTHEPARTY

Matrix for columnar encryption:
M E E T M
E A F T E
R T H E P
A R T Y *
Cipher Text: EFHTTTEYEATRMERAMEP

Matrix for columnar decryption:
M E E T M
E A F T E
R T H E P
A R T Y *
Original Text: MEETMEAFTERTHEPARTY
```

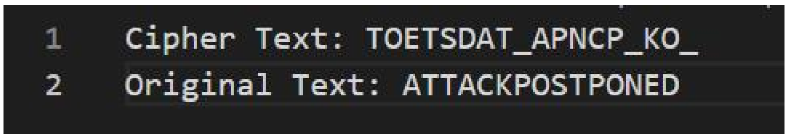
- **Sample Output 2 -**

Input file -



```
ATTACKPOSTPONED
```

Output File -



```
Cipher Text: TOETSDAT_APNCP_KO_
Original Text: ATTACKPOSTPONED
```

- **Conclusion -**

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. In case of columnar cipher , the message is written out in rows of a fixed length, and then read out again column by column, and the columns are chosen in some scrambled order. Both of them are simple transposition ciphers.