Cryptography & Network Security

PRN - 2019BTECS00026

Name - Niraja Vasudev Kulkarni

Batch - B1

Assignment - 1

- **Title - Caeser Cipher**
- **Objective -**

Decrypting the cipher text encrypted using Caesar Cipher

- **Theory** -

The Caesar Cipher technique is one of the earliest and simplest methods of encryption technique. It's simply a type of substitution cipher, i.e., each letter of a given text is replaced by a letter with a fixed number of positions down the alphabet. The method is apparently named after Julius Caesar, who apparently used it to communicate with his officials.

Procedure -

1. Take the plain text from user as an input

2. Apply the given shift & find the cipher text - For example with a shift of 3, A would be replaced by D, B would become E, and so on.

- **Code –**

```
#include <bits/stdc++.h>

using namespace std;


string encrypt(string plainText, int key)

{

    string ans = "";

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

    {

        ans += char(int(plainText[i] + key - 'A') % 26 + 'A');
```

```
    }
    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;
    int key;
    cout << "Enter Key (Shift): ";
    cin >> key;
    switch (option)
    {
    case 1:
        cout << "Enter the plain text: ";
        break;
    case 2:
        freopen("input.txt", "r", stdin);
        freopen("output.txt", "w", stdout);
        break;
    default:
        break;
    }
    cin >> plainText;
    cout << "Cipher Text: "
         << encrypt(plainText, key) << "\n";
    return 0;
}
```

- **Outputs –**

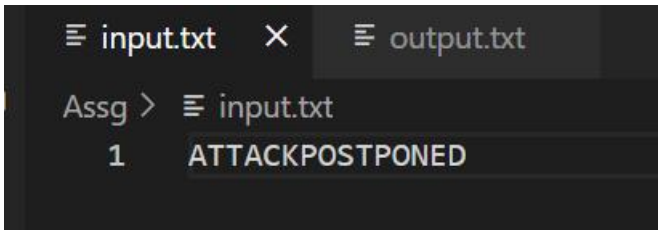
Sample output 1 –

```
D:\BTECH\CNS_LAB>cd "d:\BTECH\CNS_LAB\Assg\" && g++ CaeserCipher.cpp -o CaeserCipher && "d:\BTECH\CNS_LAB\As
sg\"CaeserCipher
How do you want to give input?:
1) Through terminal
2) Through File
1
Enter Key (Shift): 3
Enter the plain text: MEETMEAFTERTHEPARTY
Cipher Text: PHHWPHDIWHUWKHSDUWB
```

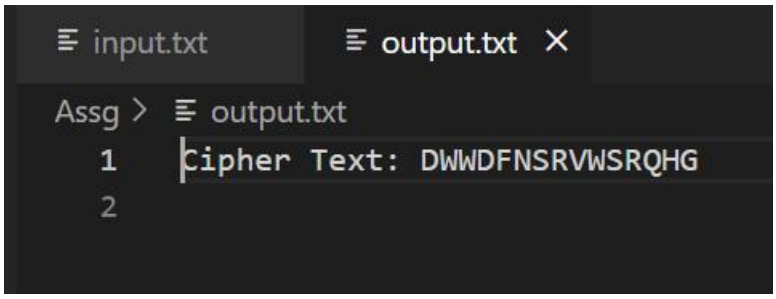
Sample output 2 -

```
d:\BTECH\CNS_LAB\Assg>cd "d:\BTECH\CNS_LAB\Assg\" && g++ CaeserCipher.cpp -o CaeserCipher && "d:\BTECH\CNS_L
AB\Assg\"CaeserCipher
How do you want to give input?:
1) Through terminal
2) Through File
2
Enter Key (Shift): 3
```

Input file -



Output file -



- **Conclusion –**

Caesar Cipher is simple substitution (mono alphabetic) technique. The key can be deciphered easily. So, it is a classical way of cryptography which is not secure in modern era.

Cryptography & Network Security

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

Assignment - 2

- **Title - Cryptanalysis (Decryption of Caeser Cipher)**

- **Objective -**

Decrypting the cipher text encrypted using Caesar Cipher

- **Theory** -

Cryptanalysis is the decryption and analysis of codes, ciphers or encrypted text. Here , the task is to perform cryptanalysis that is , to decrypt the cipher text which is actually encrypted by the Caeser Cipher . We can write another function decrypt that'll apply the shift in the opposite direction to decrypt the original text. The shift is not known , so we will have to try all possible combinations and find out which one gives meaningful output.

Procedure -

1. Take the cipher text as an input from the user
2. Considering all possible 26 shifts , decrypt the given text by applying shift in opposite direction
3. Find the meaningful output using PyEnchant library that returns whether the translated word is present in the dictionary or not

- **Code snapshots -**

```
import enchant
d = enchant.Dict("en_US")
message = input('Enter Cipher text')
LETTERS = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
actualText=''
actualKey=0
no_of_words=len(message.split())
words=[]
for key in range(len(LETTERS)):
  translated = ''
  word=''
  for symbol in message:
    if symbol in LETTERS:
      num = LETTERS.find(symbol)
      num = num - key
      if num < 0:
        num = num + len(LETTERS)
      translated = translated + LETTERS[num]
```

```
      word=word+LETTERS[num]
    else:
      if d.check(word):
         words.append(word)
      else:
         words.clear()
      word=''
  if d.check(word):
      words.append(word)
  if len(words)==no_of_words:
      actualText=translated
      actualKey=key
      words.clear()
   print('Plain text with %s: %s' % (key, translated))
print('Actual plain text is %s with key:%s' % (actualKey,actualText))
```

- **Outputs -**
- **Sample Output 1 -**

```
D:\BTECH\CNS_LAB>python -u "d:\BTECH\CNS_LAB\Cryptanalysis.py"
Enter Cipher textQIIX QI EJXIV XLI TEVXC
Plain text with 0: QIIXQIEJXIVXLITEVXC
Plain text with 1: PHHWPHDIWHUWKHSDUWB
Plain text with 2: OGGVOGCHVGTVJGRCTVA
Plain text with 3: NFFUNFBGUFSUIFQBSUZ
Plain text with 4: MEETMEAFTERTHEPARTY
Plain text with 5: LDDSLDZESDQSGDOZQSX
Plain text with 6: KCCRKCYDRCPRFCNYPRW
Plain text with 7: JBBQJBXCQBOQEBMXOQV
Plain text with 8: IAAPIAWBPANPDALWNPU
Plain text with 9: HZZOHZVAOZMOCZKVMOT
Plain text with 10: GYYNGYUZNYLNBYJULNS
Plain text with 11: FXXMFXTYMXKMAXITKMR
Plain text with 12: EWWLEWSXLWJLZWHSJLQ
Plain text with 13: DVVKDVRWKVIKYVGRIKP
Plain text with 14: CUUJCUQVJUHJXUFQHJO
Plain text with 15: BTTIBTPUITGIWTEPGIN
Plain text with 16: ASSHASOTHSFHVSDOFHM
Plain text with 17: ZRRGZRNSGREGURCNEGL
Plain text with 18: YQQFYQMRFQDFTQBMDFK
Plain text with 19: XPPEXPLQEPCESPALCEJ
Plain text with 20: WOODWOKPDOBDROZKBDI
Plain text with 21: VNNCVNJOCNACQNYJACH
Plain text with 22: UMMBUMINBMZBPMXIZBG
Plain text with 23: TLLATLHMALYAOLWHYAF
Plain text with 24: SKKZSKGLZKXZNKVGXZE
Plain text with 25: RJJYRJFKYJWYMJUFWYD
Actual plain text is 4 with key:MEETMEAFTERTHEPARTY
```

- Sample Output 2 -

```
D:\BTECH\CNS_LAB>python -u "d:\BTECH\CNS_LAB\Cryptanalysis.py"
Enter Cipher textEXXEGO TSWXTSRIH
Plain text with 0: EXXEGOTSWXTSRIH
Plain text with 1: DWWDFNSRVWSRQHG
Plain text with 2: CVVCEMRQUVRQPGF
Plain text with 3: BUUBDLQPTUQPOFE
Plain text with 4: ATTACKPOSTPONED
Plain text with 5: ZSSZBJONRSONMDC
Plain text with 6: YRRYAINMQRNMLCB
Plain text with 7: XQQXZHMLPQMLKBA
Plain text with 8: WPPWYGLKOPLKJAZ
Plain text with 9: VOOVXFKJNOKJIZY
Plain text with 10: UNNUWEJIMNJIHYX
Plain text with 11: TMMTVDIHLMIHGXW
Plain text with 12: SLLSUCHGKLHGFWV
Plain text with 13: RKKRTBGFJKGFEVU
Plain text with 14: QJJQSAFEIJFEDUT
Plain text with 15: PIIPRZEDHIEDCTS
Plain text with 16: OHHOQYDCGHDCBSR
Plain text with 17: NGGNPXCBFGCBARQ
Plain text with 18: MFFMOWBAEFBAZQP
Plain text with 19: LEELNVAZDEAZYPO
Plain text with 20: KDDKMUZYCDZYXON
Plain text with 21: JCCJLTYXBCYXWNM
Plain text with 22: IBBIKSXWABXWVML
Plain text with 23: HAAHJRWVZAWVULK
Plain text with 24: GZZGIQVUYZVUTKJ
Plain text with 25: FYYFHPUTXYUTSJI
Actual plain text is 4 with key:ATTACKPOSTPONED
```

- **Conclusion** -

Caeser Cipher is a monoalphabetic classical cipher which can be easily decrptyed. It is a naive way of encrypting. Here , PyEnchant lilbrary is used for finding the meaningful output from suggested set of sentences as the shift is unknown.

Cryptography & Network Security

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

Assignment - 3

Playfair Cipher

- **Objective -**

  To implement Playfair Cipher in C

- **Theory -**

The Playfair cipher was the first practical digraph substitution cipher. The scheme was invented in 1854 by Charles Wheatstone but was named after Lord Playfair who promoted the use of the cipher. In Playfair cipher unlike traditional cipher we encrypt a pair of alphabets(digraphs) instead of a single alphabet.
Procedure -

1. Take the plaintext & the key as input

2. Arrange the key in 5*5 matrix without repetition of characters. Fill rest of the matrix with remaining alphabets serially (Skip I/J)

3. Prepare chunks of size 2 in plaintext. If 2 characters in the chunk are same , put a dummy character.

4. Make sure the plaintext length is even otherwise put a dummy character in the end

5. For each chunk , encrypt the text by using playfair matrix with the help of following rules :

A) If both of them are in same row , take the next characters in the same row (consider wraparound)

B) If both of them are in same column , take the next characters in the same column (consider wraparound)

C) Otherwise take their intersection

Code –

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

```
int RemoveSpaces(char arr[], int len)
{
    int i, count = 0;
    for (i = 0; i < len; i++)
        if (arr[i] != ' ')
            arr[count++] = arr[i];
    arr[count] = '\0';
    return count;
}

void GenerateKeyTable(char keyarr[5], int keylen, char keyTable[5][5])
{
    int i, j, k, flag = 0;
    int map[26] = {0};
    for (i = 0; i < keylen; i++)
    {
        if (keyarr[i] != 'J')
            map[keyarr[i] - 65] = 2;
    }
    map['J' - 65] = 1;
    i = 0;
    j = 0;
    for (k = 0; k < keylen; k++)
    {
        if (map[keyarr[k] - 65] == 2)
        {
            map[keyarr[k] - 65] -= 1;
            keyTable[i][j] = keyarr[k];
            j++;
            if (j == 5)
            {
                i++;
                j = 0;
            }
        }
    }
    for (k = 0; k < 26; k++)
    {
        if (map[k] == 0)
        {
            keyTable[i][j] = (char)(k + 65);
            j++;
            if (j == 5)
            {
                i++;
                j = 0;
            }
        }
```

```
    }
}

void search(char keyTable[5][5], char a, char b, int arr[])
{
    int i, j;
    if (a == 'J')
        a = 'I';
    else if (b == 'J')
        b = 'I';
    for (i = 0; i < 5; i++)
    {
        for (j = 0; j < 5; j++)
        {
            if (keyTable[i][j] == a)
            {
                arr[0] = i;
                arr[1] = j;
            }
            else if (keyTable[i][j] == b)
            {
                arr[2] = i;
                arr[3] = j;
            }
        }
    }
}

int mod5(int a)
{
    if (a < 0)
        a += 5;
    return (a % 5);
}

int Prepare(char arr[], int len)
{
    string res;
    int i = 0;
    while (i < len - 1)
    {
        res.push_back(arr[i]);
        if (arr[i] == arr[i + 1])
        {
            res.push_back('X');
            i++;
        }
        else
```

```
        {
            res.push_back(arr[i + 1]);
            i += 2;
        }
    }
    len = res.size();
    strcpy(arr, res.c_str());
    if (len % 2 != 0)
    {
        arr[len++] = 'X';
        arr[len] = '\0';
    }
    return len;
}

void Encrypt(char textarr[], char keyTable[5][5], int textlen)
{
    int i, arr[4];
    for (i = 0; i < textlen; i += 2)
    {
        search(keyTable, textarr[i], textarr[i + 1], arr);
        if (arr[0] == arr[2])
        {
            textarr[i] = keyTable[arr[0]][mod5(arr[1] + 1)];
            textarr[i + 1] = keyTable[arr[0]][mod5(arr[3] + 1)];
        }
        else if (arr[1] == arr[3])
        {
            textarr[i] = keyTable[mod5(arr[0] + 1)][arr[1]];
            textarr[i + 1] = keyTable[mod5(arr[2] + 1)][arr[1]];
        }
        else
        {
            textarr[i] = keyTable[arr[0]][arr[3]];
            textarr[i + 1] = keyTable[arr[2]][arr[1]];
        }
    }
}

void Decrypt(char textarr[], char keyTable[5][5], int textlen)
{
    string res;
    int i, arr[4];
    for (i = 0; i < textlen; i += 2)
    {
        search(keyTable, textarr[i], textarr[i + 1], arr);
        if (arr[0] == arr[2])
        {
```

```
            textarr[i] = keyTable[arr[0]][mod5(arr[1] - 1)];
            textarr[i + 1] = keyTable[arr[0]][mod5(arr[3] - 1)];
        }
        else if (arr[1] == arr[3])
        {
            textarr[i] = keyTable[mod5(arr[0] - 1)][arr[1]];
            textarr[i + 1] = keyTable[mod5(arr[2] - 1)][arr[1]];
        }
        else
        {
            textarr[i] = keyTable[arr[0]][arr[3]];
            textarr[i + 1] = keyTable[arr[2]][arr[1]];
        }
    }
    for (i = 0; i < textlen; i++)
    {
        if(textarr[i]!='X') res.push_back(textarr[i]);
    }
    strcpy(textarr, res.c_str());
}

string PlayfairEncrypt(char textarr[], char keyarr[],int flg)
{
    char keyTable[5][5];
    int i,j;
    string result;
    int keylen = RemoveSpaces(keyarr, strlen(keyarr));
    int textlen = RemoveSpaces(textarr, strlen(textarr));
    textlen = Prepare(textarr, textlen);
    GenerateKeyTable(keyarr, keylen, keyTable);
    if(flg==1) {
        for(i = 0; i < 5; i++) {
            for(j = 0; j < 5; j++) {
                cout<<keyTable[ i ] [ j ]<<" ";
            }
            cout<<"\n";
        }
    }
    Encrypt(textarr, keyTable, textlen);
    result = textarr;
    return result;
}

string PlayfairDecrypt(char textarr[], char keyarr[])
{
    char keyTable[5][5];
    string result;
    int keylen = strlen(keyarr);
```

```
    int textlen = strlen(textarr);
    GenerateKeyTable(keyarr, keylen, keyTable);
    Decrypt(textarr, keyTable, textlen);
    result = textarr;
    return result;
}

int main()
{
    int option,flg=0;
    string key, text, cipherText;
    char keyarr[30], textarr[30], cipher[30];
    cout << "\nGive input through:\n1)Console\n2)File\n-->";
    cin >> option;
    cin.ignore();
    cout << "Enter key: ";
    getline(cin, key);
    strcpy(keyarr, key.c_str());
    switch (option)
    {
        case 1:
            cout << "Enter text: ";
            flg=1;
            break;
        case 2:
            freopen("input.txt", "r", stdin);
            freopen("output.txt", "w", stdout);
            flg=2;
            break;
    }
    getline(cin, text);
    strcpy(textarr, text.c_str());
    cipherText = PlayfairEncrypt(textarr, keyarr,flg);
    cout << "Cipher Text: " << cipherText << endl;
    strcpy(cipher, cipherText.c_str());
    cout << "Deciphered Text: " << PlayfairDecrypt(cipher, keyarr);
    return 0;
}
```

Output –

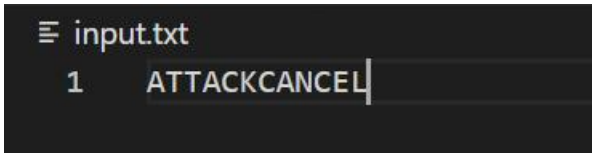
Sample output 1 –

```
D:\BTECH\CNS_LAB\Assg>cd "d:\BTECH\CNS_LAB\Assg\" && g++ PlayfairCipher.cpp -o PlayfairCipher && "d:\BTECH\CNS_LAB\Assg\"Playf
airCipher
How do you want to give input?:
1) Through terminal
2) Through File
1
Enter key: MONARCHY
Enter text: MEETMEAFTERTHEPARTY

Playfair Matrix:
M O N A R
C H Y B D
E F G I K
L P Q S T
U V W X Z
Cipher Text: CLKLCLOILKDZCFSODZ
```

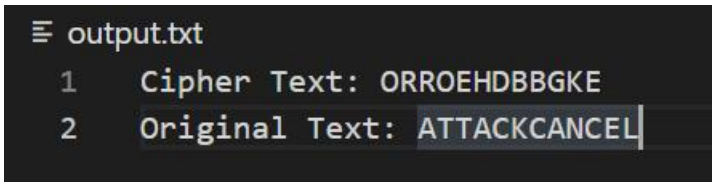
Sample output 2 -

```
D:\BTECH\CNS_LAB\Assg>cd "d:\BTECH\CNS_LAB\Assg\" && g++ PlayfairCipher.cpp -o PlayfairCipher && "d:\BTECH\CNS_LAB\Assg\"Playf
airCipher
How do you want to give input?:
1) Through terminal
2) Through File
2
Enter key: BALLOON

D:\BTECH\CNS_LAB\Assg>
```

Input file



Output file



Conclusion –

Playfair cipher is a polyalphabetic substitution cipher . Due to its easy decrption , it is not used in the modern encryption.

Cryptography & Network Security

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

Assignment - 4

Vigenere Cipher

- **Objective -**

  To implement Vigenere Cipher in C

- **Theory -**

Vigenere Cipher is a simple form of polyalphabetic substitution. A polyalphabetic cipher is any cipher based on substitution, using multiple substitution alphabets. The encryption of the original text is done using the Vigenere table. The table consists of the alphabets written out 26 times in different rows, each alphabet shifted cyclically to the left compared to the previous alphabet, corresponding to the 26 possible Caesar Ciphers. At different points in the encryption process, the cipher uses a different alphabet from one of the rows. The alphabet used at each point depends on a repeating keyword.

Procedure -

1. Take the input - plaintext & the key from user
2. Make the key size same as text size by repeating it as required
3. Add each character of key & text and find out the cipher text accordingly

- **Code Snapshots -**

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

string generateKey(string plainText, string key)
{
    int i = 0;
    while (key.size() != plainText.size())
    {
        key.push_back(key[i]);
        i++;
    }

    return key;
}
```

```
string encrypt(string plainText, string key)
{
```

```
    string ans;
    for (int i = 0; i < plainText.size(); i++)
    {
        char x = (plainText[i] + key[i]) % 26;
        x += 'A';
        ans.push_back(x);
    }
    return ans;
}
```

```
string decrypt(string cipherText, string key)
{
    string plainText;
    for (int i = 0; i < cipherText.size(); i++)
    {
        char x = (cipherText[i] - key[i] + 26) % 26;
        x += 'A';
        plainText.push_back(x);
    }
    return plainText;
}
```

```
int main()
{
    int option;
    string key, plainText, ans;
    cout << "How do you want to give input?:\n1) Through terminal\n2) Through File\n";
    cin >> option;
    cout << "Enter key: ";
    cin >> key;
    switch (option)
    {
    case 1:
        cout << "Enter plainText: ";
        break;
    case 2:
        freopen("input.txt", "r", stdin);
        freopen("output.txt", "w", stdout);
        break;
    }
    cin >> plainText;
    key = generateKey(plainText, key);
    ans = encrypt(plainText, key);
    cout << "Cipher plainText: " << ans << endl;
    cout << "Original plainText: " << decrypt(ans, key);
    return 0;
}
```

- **Outputs -**
- **Sample Output 1 -**

```
d:\BTECH\CNS_LAB\C&NS 1-6\5 - Rail Fence & Columnar Transposition Cipher\Columnar Transposition Cipher>cd "d
:\BTECH\CNS_LAB\C&NS 1-6\4 - Vigenère Cipher\" && g++ VigenereCipher.cpp -o VigenereCipher && "d:\BTECH\CNS_
LAB\C&NS 1-6\4 - Vigenère Cipher\"VigenereCipher
Enter option:
1)Console
2)File
1
Enter key: MONARCHY
Enter text: MEETMEAFTERTHEPARTY
Cipher Text: YSRTDGHDFSETYGWYDHL
Original Text: MEETMEAFTERTHEPARTY
```

- **Sample Output 2-**

```
d:\BTECH\CNS_LAB\C&NS 1-6\4 - Vigenère Cipher>cd "d:\BTECH\CNS_LAB\C&NS 1-6\4 - Vigenère Cipher\" && g++ Vig
enereCipher.cpp -o VigenereCipher && "d:\BTECH\CNS_LAB\C&NS 1-6\4 - Vigenère Cipher\"VigenereCipher
Enter option:
1)Console
2)File
2
Enter key: MONARCHY
```

Input file -

```
1    ATTACKPOSTPONED
```

Output file -

```
1    Cipher Text: MHGATMWMEHCOEGK
2    Original Text: ATTACKPOSTPONED
```

- **Conclusion -**

Vigenere Cipher is polyalphabetic substitution cipher , in which a single alphabet can be encrypted with different alphabets when its occurrence is repeated.

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++];
        }
```

```
    }
    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 -

```
1   ATTACKPOSTPONED
```

Output file -

```
1   Cipher Text: ACSNTAKOTOETPPD
2   Original Text: ATTACKPOSTPONED
```

2. Columar Cipher -

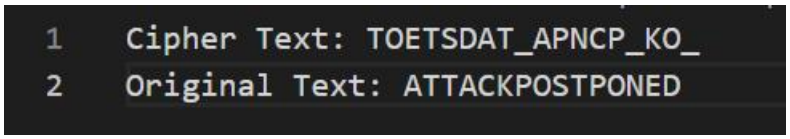
- Sample Output 1 -

```
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 -

Output File -



- **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.