21.08.2020

Ex No: 2.B **IMPLEMENTATION OF ROW & COLUMN TRANSFORMATION TECHNIQUE**

AIM:

To write a C program to implement the row and column transformation technique

DESCRIPTION:

In the row and column transformation, a plain-text message and a numeric key, cipher/de-cipher the given text using Columnar Transposition Cipher. 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.

ALGORITHM:

STEP-1: Read the Plaintext and apply the simple columnar transposition technique

STEP-2: The plain text characters are placed horizontally and the cipher text is created with vertical format

STEP-3: Now, the receiver has to use the same table to decrypt the cipher text to plain text.

STEP-4: Arrange the characters of the keyword in sorted order and the corresponding columns of the plain text.

STEP-5: Read the characters row wise or column wise in the former order to get the cipher text.

PROGRAM:

```
// CPP program for illustrating
// Columnar Transposition Cipher
#include<bits/stdc++.h>
using namespace std;

// Key for Columnar Transposition
string const key = "HACK";
map<int,int> keyMap;

void setPermutationOrder()
{
        // Add the permutation order into map
        for(int i=0; i < key.length(); i++)
        {
             keyMap[key[i]] = i;
        }
}

// Encryption
string encryptMessage(string msg)
{
        int row,col,j;
        string cipher = "";

        /* calculate column of the matrix*/
```

```
col = key.length();

/* calculate Maximum row of the matrix*/
row = msg.length()/col;

if (msg.length() % col)
     row += 1;

char matrix[row][col];

for (int i=0,k=0; i < row; i++)
{
     for (int j=0; j<col; )
     {
           if(msg[k] == '\0')
           {
                 /* Adding the padding character '_' */
                 matrix[i][j] = '_';
                 j++;
           }

           if( isalpha(msg[k]) || msg[k]==' ')
           {
                 /* Adding only space and alphabet into matrix*/
                 matrix[i][j] = msg[k];
                 j++;
           }
           k++;
     }
```

```
        }

        for (map<int,int>::iterator ii = keyMap.begin(); ii!=keyMap.end();
++ii)
        {
             j=ii->second;

             // getting cipher text from matrix column wise using permuted key
             for (int i=0; i<row; i++)
             {
                   if( isalpha(matrix[i][j]) || matrix[i][j]==' ' ||
matrix[i][j]=='_')
                         cipher += matrix[i][j];
             }
        }

        return cipher;
}

// Decryption
string decryptMessage(string cipher)
{
        /* calculate row and column for cipher Matrix */
        int col = key.length();

        int row = cipher.length()/col;
        char cipherMat[row][col];

        /* add character into matrix column wise */
        for (int j=0,k=0; j<col; j++)
```

```
        for (int i=0; i<row; i++)
                cipherMat[i][j] = cipher[k++];

    /* update the order of key for decryption */
    int index = 0;
    for( map<int,int>::iterator ii=keyMap.begin(); ii!=keyMap.end(); ++ii)
        ii->second = index++;

    /* Arrange the matrix column wise according
    to permutation order by adding into new matrix */
    char decCipher[row][col];
    map<int,int>::iterator ii=keyMap.begin();
    int k = 0;
    for (int l=0,j; key[l]!='\0'; k++)
    {
        j = keyMap[key[l++]];
        for (int i=0; i<row; i++)
        {
                decCipher[i][k]=cipherMat[i][j];
        }
    }

    /* getting Message using matrix */
    string msg = "";
    for (int i=0; i<row; i++)
    {
        for(int j=0; j<col; j++)
        {
                if(decCipher[i][j] != '_')
```

```
                        msg += decCipher[i][j];
                }
        }
        return msg;
}

// Driver Program
int main(void)
{
        /* message */
        string msg = "Geedha.india";

        setPermutationOrder();

        // Calling encryption function
        string cipher = encryptMessage(msg);
        cout << "Encrypted Message: " << cipher << endl;

        // Calling Decryption function
        cout << "Decrypted Message: " << decryptMessage(cipher) << endl;

        return 0;
}
```

OUTPUT:

```
Encrypted Message: eaieiaGhddn_
Decrypted Message: Geedhaindia
```

RESULT:

Thus the transposition technique algorithm had been executed successfully.

.