Lab-06

AIM:- Write a program to implement Hill cipher for 2x2, and 3x3 matrics.

Source code:-

```
#include "functions.h"
#define matrix vector<vector<int>>

int domod26(int num)
{
    while (num < 26)
        num += 26;
    return num % 26;
}
matrix getCofactor(matrix, mat, int p, int q)
{
    int i = 0, j = 0, n = mat.size();
    matrix cofactorMat (n-1, vector<int>(n-1));

    for(int col = 0; col < n; ++col)
        if (row != p && col != q)
        {
            cofactorMat[i][j++] = mat[row][col];
            if( j == n-1)
            {   j = 0;
                i++
```

```
    }
  }

  return cofactorMat;

int determinant(matrix mat, int size)
{
    int n = mat.size(); d=0;
    if (size == 1)
        return mat[0][0];
    int sign=1;
    matrix cofactorMat(n, vector<int>(n));
    for(col=0; col<size; ++col)
    {
        cofactorMat = getCofactor(mat, 0, col);

        d += sign * mat[0][col] *
                determinant(cofactorMat,
                            n-1);

        sign = -1 * sign;
    }

    return d%26(d);
}
```

```
matrix adjointMatrix(matrix mat)
{
    int n = mat.size();
    if(n == 1)
    {
        mat[0][0] = 1;
        return mat;
    }
    int sign = 1;

    for(int i=0; i<n; ++i)
        for(int j=0; j<n; ++j)
        {
            cofactorMat = getCofactor(mat, i, j);
            sign = (i+j)%2 == 0 ? 1 : -1;
            adjointMat[j][i] = sign *
                                determinant(
                                cofactorMat, n-1);
            adjoint mat[j][i] = domod 26
                                (adjointMat[j][i]

        }

    return adjoint Mat;
}
```

```
matrix findInverseMatrix(matrix mat)
{
    int n = mat.size(),
        inverseDet = multiplicativeInverse
                     (determinant(mat, n), 26);

    adjointMat = adjointMatrix(mat);

    for(int i=0; i<n; ++i)
        for(int j=0; j<n; ++j)
        {
            inverseMat[i][j]
                = domod26(adjointMat[i][j]
                          * inverseDet)];
        }
    return inverseDet;
}

matrix generateKeyMatrix(int size)
{
    matrix keyMatrix;
    if (size==2)
    {
        matrix key{{7,11},{8,11}};
        keyMatrix = key;
    }
```

```
else if (size==3)
{
    matrix key{{7,5,11},{4,18,1},
                {3,18,1}};
    keyMatrix = key;
}
else if (size==4)
{
    matrix key{ {9,7,11,13},
                {4,7,5,6},
                {2,21,14,9}
                {3,23,21,8}};

    keyMatrix =key;
}

return keyMatrix;
}

void printMatrix(matrix mat)
{
    for(int i=0; i<mat.size; ++i)
    {   for(int j=0; j<mat[0].size(); ++j)
            cout<<mat[i][j]<<" ";
    cout<<endl;
    }
```

```
}

matrix generatePlainTextMatrix
        (String plainText, int column).
{
    int row = ceil(plainText.length()*1.0 /
                                column);

    if(trow * column) > plainText.length())
    {
        int ditt = row* column - plainText.
                                lenght()

        while(ditt--)
            plainText += 'z';
    }

    for(int i=0; i<row; ++i)
        for(int j=0; j<column; ++j)
            plainTextmetrix[i][j]
            = int(plainText[column * i+j]-'a'

    return plainTextmatrix;
}
```

```
matrix encryption(matrix key, matrix plainText
                                        matrix)
{
    int row = plainTextMatrix.size(),
        column = plainTextMatrix[0].size();

    matrix cipherTextmatrix;
    int temp;
    for(int i=0; i<row; ++i)
        for(int j=0; j<column; ++j)
        {   for(int k=0; k<column; ++k)
                temp += plainTextmatrix[i][j]
                        * key[i][j]  %26

            cipherTextMatrix[i][j] = temp%26;
            temp=0;
        }

    return cipherTextMatrix;
}
string convertToText(matrix mat)
{
    string text = "";
    for(int i=0; i<mat.size(); ++i)
        for(int j=0; j<mat[0].size(); ++j)
            text += mat[j][i] + 'a';
    return text;
}
```

```
String decryption (matrix ciphor, matrix key)
{
    inverseMat = findInverseMatrix( key );
    int temp=0;
    for(int i=0; i<row; ++i)
    {
        for(int j=0; j<n; ++j)
        {
            for(int k=0; k<n; ++k)
            {
                temp += ciphorTextMatrix[i][k]
                        * inverTestMat[k][j];
            }
            decryptedMatrix[i][j] = domod(temp);
            temp=0;
        }
    cout<< "Decrypted Matrix: \n";
    printMatrix( decryptedMatrix);
    return convertToText(decryptedMatrix);
}

int main()
{
    string plainText, decryptedText;
    int size;
    cout<<" Enter plainText : ";
```

```
cin>>plainText;
cout<<"Enter size of the key matrix: ";
cin>>size;

if(size>4 || size<2){
    cout<<"size is not valid";
    exit(0);
}

keyMatrix = generateKeyMatrix(size);
ptMatrix = generatePlainTextMatrix(plainText,
                                   size);

cout<<"key matrix:\n";
printMatrix(keyMatrix);
cout<<"plain Text matrix:"<<endl;
printMatrix(plainTextMatrix);
cipherTextMatrix = encryption(keyMatrix,
                              plainTextMatrix);
cout<<"Encryption matrix:\n";
print(cipherText matrix.");

cout<<"Encrypted Text:"<<convertToText(
                              cipherTextMatrix)<<endl;
decryptedText = decryption(cipherTextMatrix,
                           keyMatrix);
cout<<"Decrypted Text:"<<decryptedText<<endl;
return 0;
}
```

Input

Enter plainText: munaf divan

Enter size of the key matrix: 3

o/p

Key matrix:

7	25	11
4	18	1
3	18	1

plain Text matrix:

12	20	13
0	5	3
8	21	0
13	25	25

Encryption matrix:

21	10	9
3	14	8
10	6	5
6	3	11

Encrypted Text: vkjdojkgfgdl

Decrypted matrix

12	20	13
0	5	3
8	21	0
13	25	25

Decrypted Text: munafdivanzz.