LAB - 6

Q1) Aim: Write a program to implement encryption and decryption using hill cipher for 2×2 and 3×3 matrices.

Ans Code

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

vector<vector<int>> getAdjoint (vector<vector<int>> matrix)
{
    vector<vector<int>> retMat {{0,0,0},{0,0,0},{0,0,0}};

    retMat[0][0] = (matrix[1][1] * matrix[2][2]) - (matrix[1][2]
                                    * matrix[2][0]));

    retMat[1][0] = -1 * ((matrix[1][0] * matrix[2][2]) -
                        (matrix[1][2] * matrix[2][0]));

    retMat[2][0] = (matrix[1][0] * matrix[2][1]) - (matrix[2][0]
                                    * matrix[1][1]);

    retMat[0][1] = -1 * ((matrix[0][1] * matrix[2][2]) -
                        (matrix[0][2] * matrix[2][1]));
    retMat[1][1] = (matrix[0][0] * matrix[2][2]) - (matrix[2][0]
                                              * matrix[0][2]);

    retmat[2][1] = -1 x ((matrix[0][0] * matrix[2][1]) -
                           (matrix[0][1] x matrix[2][0]));

     retMat[0][2] = (matrix[0][1] * matrix[1][2])
                      - (matrix[0][2] * matrix[1][0]);

      retMat[1][2] = -1* ((matrix[0][0] * matrix[1][2])
                            - (matrix[0][2] * matrix[1][0]));

      retMat[2][2] = (matrix[0][0] * matrix[1][1])
                       - (matrix[0][1] * matrix[1][0]);

       return retMat;

}

int getDeterminant (vector<vector<int>> matrix)
{
      int det = 0;
      det += (matrix[0][0]) * ((matrix[1][1] * matrix[2][2])
                                - (matrix[1][2] * matrix[2][1]));

      det += (matrix[0][1]) * ((matrix[1][0] * matrix[2][2])
                                - (matrix[1][2] * matrix[2][0]));
    det += (matrix [0][2]) * ((matrix[1](0] * matrix[2][1])
                                - (matrix[1] (1] * matrix [2] (0] ));

    return det;

}

vector<vector<int>> convertStringToMatrix ( string s, int row, int col)
{

    vector<vector<int>> mat;
    strcntr=0;
    for(int i=0 ; i< row; i++)
    {   vector<int> v;
        for (int j=0 ;j< col ; j++)
        {.
            v.push-back((s[strcntr++] -'a'] % 26);
        }
        mat.push-back(v);
    }
    return mat
}

String   convertString (string s, int matLen)
{

    int paddingchars = matlen - int(s.size() % matlen);
    if (paddingchars == matLen)
    {
        return s;
    }
    else
    {
        for (int i=0; i< paddingchars; i++)
        {
            s.push-back('z');
        }
        return s;
    }
}

void printmatrix( vector<vector<int>> matrix )
{
    for (auto x : v)
    {
        for (auto y : x)
        {
            cout<<y<<" ";
        }
        cout<<endl;
    }
}
```

```
vector<vector<int>> multiplyMatrix(vector<vector<int>> k1,
 vector<vector<int>> m)
{
        int R1, C1, R2, C2;
        R1 = m.size();
        R2 = k1.size())
        C1 = m[0].size();
        C2 = k1[0].size();

        vector<vector<int>> multipliedMatrix;

        for (int i=0; i<R1; i++)
        {    vector<int> v;
             for (int j=0; i<C2; j++)
             {
                  int res=0;
                  for (int k=0; k<R2; k++)
                  {
                       res += ((m[i][k] * k1[k][j]));
                  }
                  v.push_back (res%26);
             }
             multipliedMatrix.push_back(v);
        }
        return multipliedMatrix;
}
```

```
vector<vector<int>> encrypt ( vector<vector<int>> key,
                              vector<vector<int>> message)
{
    return multiplyMatrix ( key, message);
}

int extendedEuclidianAlgorithm( int o, int n)
{
    // Implemented in LAB-2.
}

string matrixToString ( vector<vector<int>> matrix)
{
    string retStr = " ";
    for (auto x : matrix)
    {
        for (auto y : x)
        {
            retStr += ((n + 'a');
        }
    }
    return retStr;
}
```

```
vector<vector<int>> decrypt( vector<vector<int>> key ,
                             vector<vector<int>> message )
{
    int determinant = getDeterminant (key) % 26 ;
    determinant  =  getInverse (determinant, 26);

    vector<vector<int>> adj  =  getAdjoint (key);

    for( int i=0; i< adj.size(); i++)
    {
        for(int j=0; j< adj[0].size(); j++)
        {
            adj[i][j] = (adj[i][j] * determinant) % 26;

            if(adj[i][j]<0)
            {
                adj[i][j]+=26;
            }
        }
    }

    vector<vector<int>> multipliedMatrix = multiplymatrix(adj,
                                                   message);

    return  multpliedMatrix;
}
```

```
int main()
{
    int tt;
    cin>> tt;

    while(tt--)
    {
        int matdim;
        cin>> matdim;

        string key, message;
        cin>> key >> message;

        if ( key.size()/matdim != matdim)
        {
            cout<<" Key length must be of length <<
              matdim * matdim << endl;
            exit(0);
        }
        vector<vector<int>> keyMatrix = convertStringToMatrix
                                          (key, matdim, matdim);

        string paddedMessage = convertString(message, matdim);

        vector<vector<int>> messageMatrix = convertStringTo
          Matrix (paddedMessage, paddedMessage.size/ matdim, matdim)
                                                                   ;
    cout<< "Key matrix \n";
    printMatrix( keyMatrix);

    cout<<" Message matrix \n";
    printMatrix( messageMatrix);

    vector<vector<int>> mul = encrypt( keyMatrix, messageMatrix);

    cout<<" Encrypted Matrix \n";
    printMatrix(mul);

    vector<vector<int>> decryptedMsg = decrypt (keyMatrix, mul);
    cout<<"Decrypted Matrix \n";
    printmatrix( decryptedMsg);

    cout<<"Decrypted String \n";

    cout<< matrixToString( decryptedMsg);

  }
  return 0;
}.

// End Of Code.
```

Examples of Input & Output

Input.txt

```
2
3
jhlnehfgc
code is ready
3
jeniljeni
king
```

Output.txt

Case #1

Key matrix:

```
9 7 11
13 4 7
5 6 2
```

message matrix

```
2 14 3
4 8 18
17 4 0
3 24 25
```

Encrypted Matrix

```
7 10 22
22 12 6
23 5 7
22 7 17
```

Decrypted matrix

```
2 14 3
4 8 18
17 4 0
3 24 25
```

Decrypted String: Codeisreadyz

Case #2

Key matrix:

$$\begin{matrix} 9 & 4 & 13 \\ 8 & 11 & 9 \\ 4 & 13 & 8 \end{matrix}$$

message matrix

$$\begin{matrix} 10 & 8 & 13 \\ 6 & 25 & 25 \end{matrix}$$

Encrypted matrix

$$\begin{matrix} 24 & 11 & 20 \\ 16 & 0 & 9 \end{matrix}$$

Decrypted matrix

$$\begin{matrix} 10 & 8 & 13 \\ 6 & 25 & 25 \end{matrix}$$

Decrypted String : kingzz

X—End—X