Program – 6

AIM: To implement a program to show encryption and decryption using Hill Cipher.

Introduction and Theory

Hill cipher is a polygraphic substitution cipher based on linear algebra. Each letter is represented by a number modulo 26. Often the simple scheme A = 0, B = 1, …, Z = 25 is used, but this is not an essential feature of the cipher. To encrypt a message, each block of n letters (considered as an n-component vector) is multiplied by an invertible n × n matrix, against modulus 26. To decrypt the message, each block is multiplied by the inverse of the matrix used for encryption. The matrix used for encryption is the cipher key, and it should be chosen randomly from the set of invertible n × n matrices (modulo 26)

$$Key(3x3) = \begin{matrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{matrix}$$

Encryption

In order to encrypt a message using the Hill cipher, the sender and receiver must first agree upon a key matrix A of size n x n. A must be invertible mod 26. The plaintext will then be enciphered in blocks of size n. In the following example A is a 2 x 2 matrix and the message will be enciphered in blocks of 2 characters.

$$Encrypt(\text{ACT}) = POH$$

$$\begin{bmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{bmatrix} \begin{bmatrix} 0 \\ 2 \\ 19 \end{bmatrix} = \begin{bmatrix} 67 \\ 222 \\ 319 \end{bmatrix} \equiv \begin{bmatrix} 15 \\ 14 \\ 7 \end{bmatrix} \%26$$

Decryption

Decryption follows Encryption but uses the inverse of the encryption matrix instead.

$$Decrypt(\text{"POH"}) = ACT$$

$$\begin{bmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{bmatrix}^{-1} \begin{bmatrix} 15 \\ 14 \\ 7 \end{bmatrix} = \begin{bmatrix} 260 \\ 574 \\ 539 \end{bmatrix} \equiv \begin{bmatrix} 0 \\ 2 \\ 19 \end{bmatrix} \%26$$

Program – 6

Code

```
#include<iostream>
#include<cmath>
#define K 5

using namespace std;

class HillCipher
{
public:
        HillCipher() {}
        void encrypt();
        void decrypt();
        void SetData();
        void CalculateInverse();
        float enc[K][1];
        float dec[K][1];
        float A[K][K];
        float B[K][K];
        float C[K][K];
        float message[K][1];
};

void getCofactor(float A[K][K], float temp[K][K], int p, int q, int
n)
{
    int i = 0, j = 0;

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

int determinant(float A[K][K], int n)
{
    int D = 0;
    if (n == 1)
        return A[0][0];

    float temp[K][K];

    int sign = 1;

```

Program – 6

```cpp
    for (int f = 0; f < n; f++)
    {
        getCofactor(A, temp, 0, f, n);
        D += sign * A[0][f] * determinant(temp, n - 1);

        sign = -sign;
    }

    return D;
}

void adjoint(float A[K][K], float adj[K][K])
{
    if (K == 1)
    {
        adj[0][0] = 1;
        return;
    }

    int sign = 1;
    float temp[K][K];

    for (int i = 0; i < K; i++)
    {
        for (int j = 0; j < K; j++)
        {
            getCofactor(A, temp, i, j, K);

            sign = ((i + j) % 2 == 0) ? 1 : -1;

            adj[j][i] = (sign)*(determinant(temp, K - 1));
        }
    }
}

bool inverse(float A[K][K], float inverse[K][K])
{
    float det = determinant(A, K);
    if (det == 0)
    {
        cout << "Singular matrix, can't find its inverse";
        return false;
    }

    float adj[K][K];
    adjoint(A, adj);

    for (int i = 0; i < K; i++)
        for (int j = 0; j < K; j++)
            inverse[i][j] = adj[i][j] / float(det);

    return true;
}

void HillCipher::encrypt()
{
        int i, k;
```

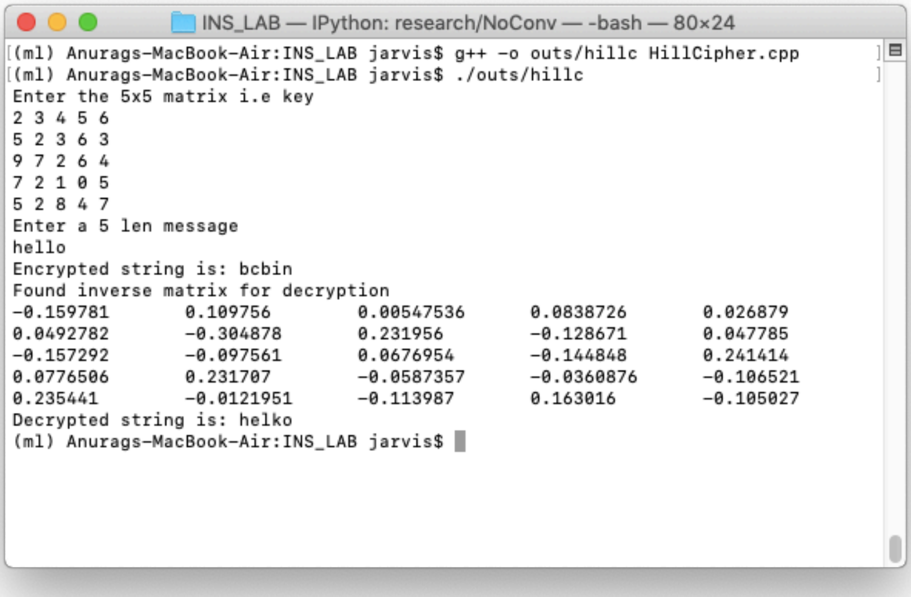
Program – 6

```cpp

        for (i = 0; i < K; i++)
                for (k = 0; k < K; k++)
                        enc[i][0] = enc[i][0] + A[i][k] *
message[k][0];

        cout << "Encrypted string is: ";
        for (int i = 0; i < K; i++)
                cout << (char)(fmod(enc[i][0], 26) + 97);
        cout << endl;
}

void HillCipher::decrypt()
{
        int i, k;
        CalculateInverse();

        for ( i = 0; i < K; i++)
                for ( k = 0; k < K; k++)
                        dec[i][0] = dec[i][0] + B[i][k] * enc[k][0];

        cout << "Decrypted string is: ";
        for (int i = 0; i < K; i++)
                cout << (char)(fmod((int)dec[i][0], 26) + 97);
        cout << endl;
}

void HillCipher::SetData()
{
        char msg[K];

        cout << "Enter the " << K << "x" << K << " matrix i.e key"
<< endl;
        for (int i = 0; i < K; i++)
        {
                for (int j = 0; j < K; j++)
                {
                        cin >> A[i][j];
                        C[i][j] = A[i][j];
                }
        }

        cout << "Enter a " << K << " len message" << endl;
        cin >> msg;
        for (int i = 0; i < K; i++)
        {
                message[i][0] = msg[i] - 97;
        }
}

void HillCipher::CalculateInverse()
{
    bool b = inverse(A, B);

        cout << "Found inverse matrix for decryption" << endl;
        for (int i = 0; i < K; i++)
        {
```

Program – 6

```
                for (int j = 0; j < K; j++)
                {
                        cout << B[i][j] << '\t';
                }
                cout << endl;
        }
}
int main()
{
        HillCipher cipher;
        cipher.SetData();
        cipher.encrypt();
        cipher.decrypt();
        return 0;
}
```

Results and Outputs:



```
(ml) Anurags-MacBook-Air:INS_LAB jarvis$ g++ -o outs/hillc HillCipher.cpp
(ml) Anurags-MacBook-Air:INS_LAB jarvis$ ./outs/hillc
Enter the 5x5 matrix i.e key
2 3 4 5 6
5 2 3 6 3
9 7 2 6 4
7 2 1 0 5
5 2 8 4 7
Enter a 5 len message
hello
Encrypted string is: bcbin
Found inverse matrix for decryption
-0.159781       0.109756        0.00547536      0.0838726       0.026879
0.0492782       -0.304878       0.231956        -0.128671       0.047785
-0.157292       -0.097561       0.0676954       -0.144848       0.241414
0.0776506       0.231707        -0.0587357      -0.0360876      -0.106521
0.235441        -0.0121951      -0.113987       0.163016        -0.105027
Decrypted string is: helko
(ml) Anurags-MacBook-Air:INS_LAB jarvis$ 
```

Findings and Learnings:

1. We have implemented Hill Cipher.