**EX.NO.2**

**HILL CIPHER – ENCRYPTION AND DECRYPTION**

**CODE:**

import java.util.Scanner;

class HillCipher {

String plainText = new String();

String cipherText = new String();

String key = new String();

int det;

int[][] key\_mat = new int[3][3];

int[][] plain\_mat = new int[3][1];

int[][] cipher\_mat = new int[3][1];

int[][] key\_inv\_mat = new int[3][3];

int[][] adj = new int[3][3];

void printPlainMat() {

System.out.println("\nThe plain text vector is ");

for (int i = 0; i < 3; i++) {

plain\_mat[i][0] = plainText.charAt(i) - 97;

System.out.println(plain\_mat[i][0]);

}

}

void printCipherMat() {

System.out.println("\nThe cipher text vector is ");

for (int i = 0; i < 3; i++) {

cipher\_mat[i][0] = cipherText.charAt(i) - 97;

System.out.println(cipher\_mat[i][0]);

}

}

void printKeyMat() {

System.out.println("\nThe key matrix is\n ");

int k = 0;

for (int i = 0; i < 3; i++) {

for (int j = 0; j < 3; j++) {

key\_mat[i][j] = (int) key.charAt(k) - 97;

System.out.print(key\_mat[i][j] + " ");

k++;

}

System.out.println();

}

}

void printKeyInverseMat(int detInv) {

System.out.println("\nThe inverse of the key matrix is\n ");

int k = 0;

for (int i = 0; i < 3; i++) {

for (int j = 0; j < 3; j++) {

key\_inv\_mat[i][j] = (adj[i][j] \* detInv) % 26;

System.out.print(key\_inv\_mat[i][j] + " ");

k++;

}

System.out.println();

}

}

void encrypt() {

System.out.println("\nThe cipher text vector is ");

for (int i = 0; i < 3; i++) {

for (int j = 0; j < 1; j++) {

cipher\_mat[i][j] = 0;

for (int k = 0; k < 3; k++) {

cipher\_mat[i][j] += key\_mat[i][k] \* plain\_mat[k][j];

}

cipher\_mat[i][j] = cipher\_mat[i][j] % 26;

System.out.print(cipher\_mat[i][j] + " ");

}

System.out.println();

}

// cipherText="";

for (int i = 0; i < 3; i++) cipherText += (char) (cipher\_mat[i][0] + 97);

System.out.println("\nThe ciphertext is: " + cipherText);

}

void decrypt() {

System.out.println("\nThe plain text vector is ");

for (int i = 0; i < 3; i++) {

for (int j = 0; j < 1; j++) {

plain\_mat[i][j] = 0;

for (int k = 0; k < 3; k++) {

plain\_mat[i][j] += key\_inv\_mat[i][k] \* cipher\_mat[k][j];

}

plain\_mat[i][j] = plain\_mat[i][j] % 26;

System.out.print(plain\_mat[i][j] + " ");

}

System.out.println();

}

plainText = "";

for (int i = 0; i < 3; i++) plainText += (char) (plain\_mat[i][0] + 97);

System.out.println("\nThe plaintext is: " + plainText);

}

public boolean isInvertible() {

det = findDet(key\_mat, 3);

// System.out.println("det is "+det);

if (det == 0 || det % 2 == 0 || det % 13 == 0) {

return false;

} else {

return true;

}

}

int modInverse(int a, int m) {

a = a % m;

for (int x = 1; x < m; x++) if ((a \* x) % m == 1) return x;

return 1;

}

void findAdjoint() {

int sign = 1;

int[][] temp = new int[3][3];

for (int i = 0; i < 3; i++) {

for (int j = 0; j < 3; j++) {

getCofactor(key\_mat, temp, i, j, 3);

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

adj[j][i] = (sign) \* (findDet(temp, 2));

while (adj[j][i] < 0) {

adj[j][i] += 26;

}

}

}

}

void getCofactor(int mat[][], int temp[][], 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++] = mat[row][col];

if (j == n - 1) {

j = 0;

i++;

}

}

}

}

}

int findDet(int mat[][], int n) {

int det = 0;

if (n == 1) return mat[0][0];

int temp[][] = new int[3][3];

int sign = 1;

for (int f = 0; f < n; f++) {

getCofactor(mat, temp, 0, f, n);

det += sign \* mat[0][f] \* findDet(temp, n - 1);

sign = -sign;

}

while (det < 0) {

det += 26;

}

return det % 26;

}

public static void main(String[] args) {

HillCipher hc = new HillCipher();

Scanner sc = new Scanner(System.in);

System.out.println("HILL CIPHER");

System.out.println("\nENCRYPTION");

System.out.println("\*\*\*\*\*\*\*\*\*\*");

System.out.print("\nEnter plain text: ");

hc.plainText = sc.next();

System.out.print("\nEnter the key: ");

hc.key = sc.next();

hc.printKeyMat();

hc.printPlainMat();

hc.encrypt();

System.out.println("\nDECRYPTION");

System.out.println("\*\*\*\*\*\*\*\*\*\*");

System.out.print("\nEnter cipher text: ");

hc.cipherText = sc.next();

System.out.print("\nEnter the key: ");

hc.key = sc.next();

hc.printKeyMat();

if (!hc.isInvertible()) System.out.println("Key is not invertible"); else {

int detInv = hc.modInverse(hc.det, 26);

hc.findAdjoint();

hc.printKeyInverseMat(detInv);

hc.printCipherMat();

hc.decrypt();

}

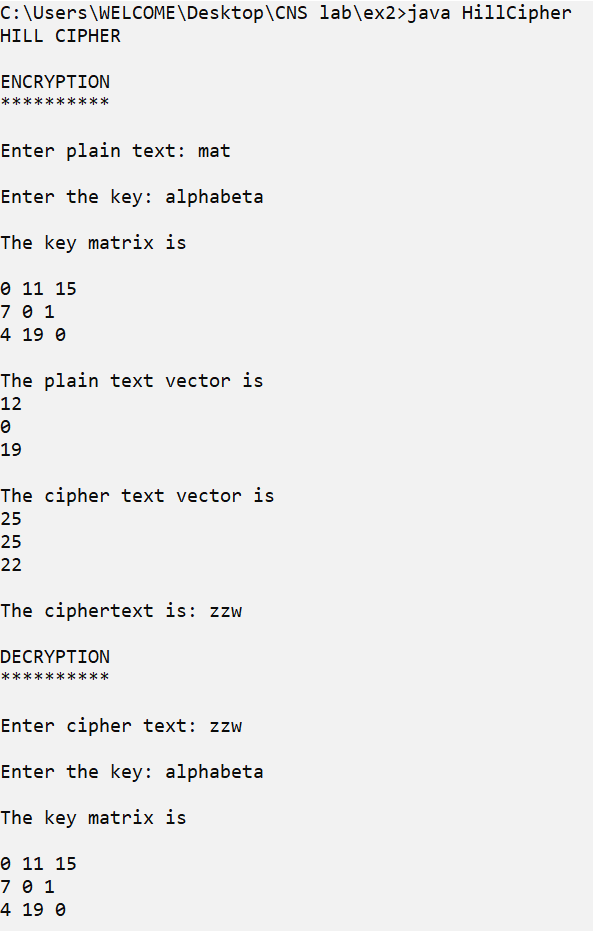
sc.close();

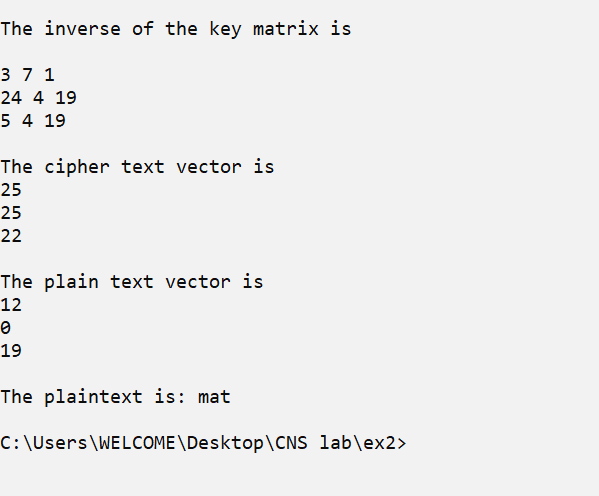
}

}

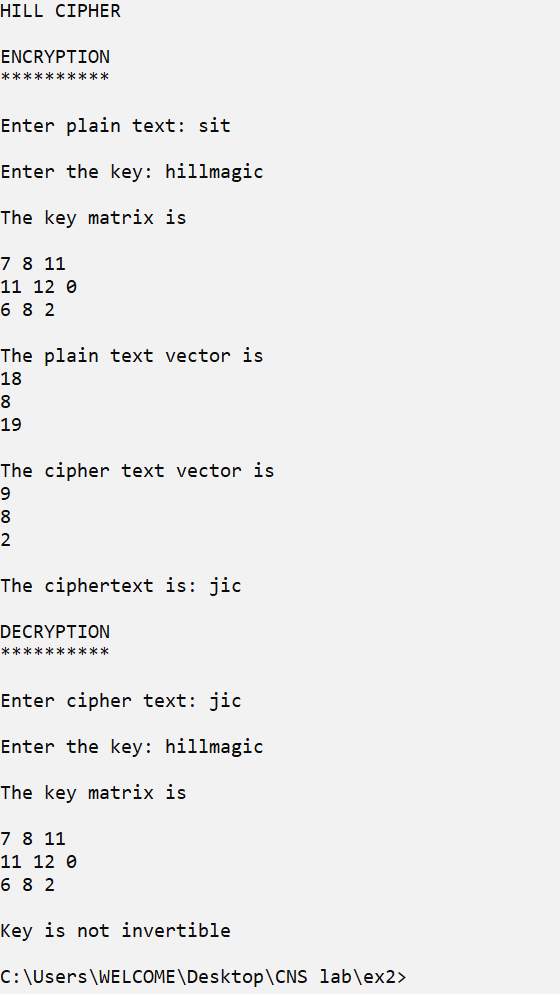
**OUTPUT:**

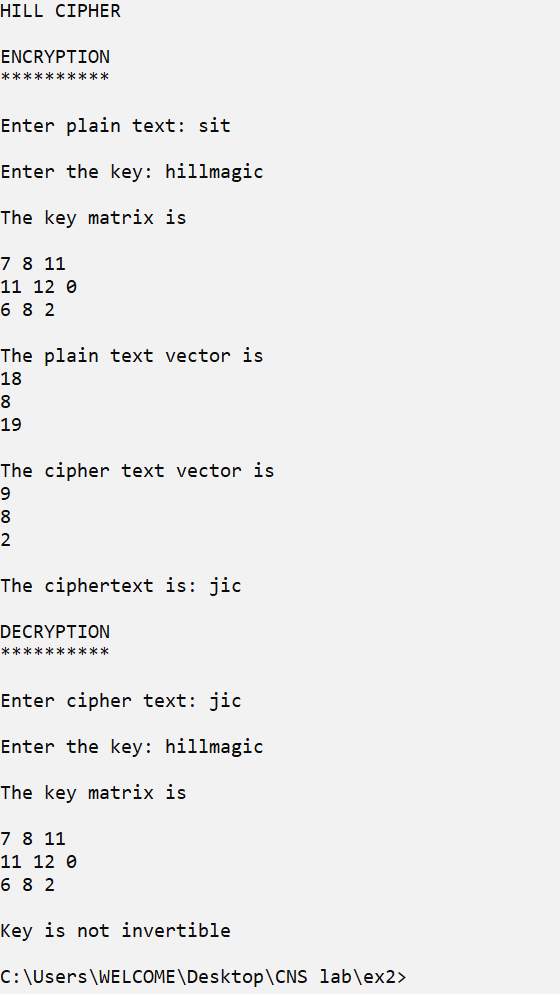
**Example 1:**





**Example 2:**

****

****

**VIGENERE CIPHER –ENCRYPTION AND DECRYTION**

**CODE:**

import java.util.Scanner;

public class VigenereCipher {

String plainText = new String();

String cipherText = new String();

String key = new String();

void generateKey() {

int diff;

while (key.length() != plainText.length()) {

diff = plainText.length() - key.length();

if (diff >= key.length())

key += key;

else

key = key + key.substring(0, diff);

}

}

void encrypt() {

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

int x = (plainText.charAt(i) + key.charAt(i)) % 26;

x += 'A';

cipherText += (char) (x);

}

}

void decrypt() {

for (int i = 0; i < cipherText.length() && i < key.length(); i++) {

int x = (cipherText.charAt(i) - key.charAt(i) + 26) % 26;

x += 'A';

plainText += (char) (x);

}

}

public static void main(String[] args) {

VigenereCipher vc = new VigenereCipher();

Scanner sc = new Scanner(System.in);

System.out.println("VIGENERE CIPHER");

System.out.println("\nENCRYPTION");

System.out.println("\*\*\*\*\*\*\*\*\*\*");

System.out.print("\nEnter plain text: ");

vc.plainText = sc.next();

System.out.print("\nEnter the key: ");

vc.key = sc.next();

vc.generateKey();

vc.encrypt();

System.out.println("\nCipher text is: " + vc.cipherText);

System.out.println("\nDECRYPTION");

System.out.println("\*\*\*\*\*\*\*\*\*\*");

System.out.print("\nEnter cipher text: ");

vc.cipherText = sc.next();

System.out.print("\nEnter the key: ");

vc.key = sc.next();

vc.generateKey();

vc.plainText = "";

vc.decrypt();

System.out.println("\nPlain text is: " + vc.plainText);

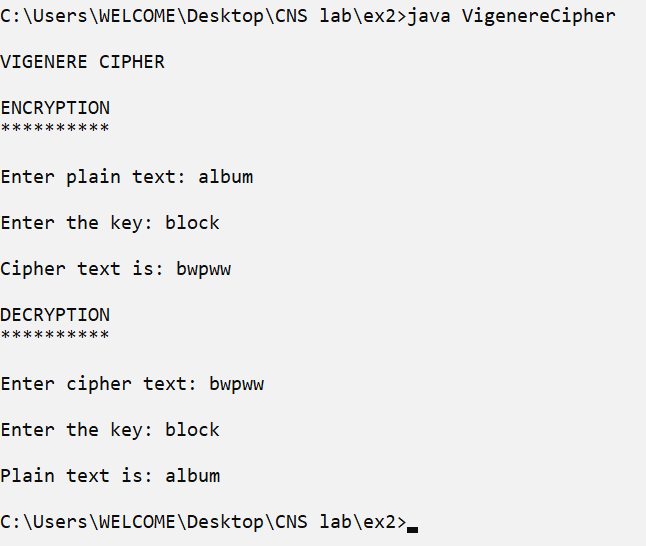
sc.close();

}

}

**OUTPUT:**

**Example 1:**

****

**Example 2:**

****

****

**Example 3:**

****