**IT8761 – Security Laboratory**

**Reshma Ramesh Babu**

**312217104129**

**Aim:** To implement the substitution ciphers: Hill cipher and Vigenere Cipher.

**Hill Cipher:**

**Code:**

import java.util.\*;

import java.io.\*;

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

public class Hill {

static int[] lm;

static int[][] keyMatrix;

static int[] rm;

static int choice;

static int [][] inverseKeyMatrix;

static int casevariable;

static String line="";

// Display function to print a matrix

public static void displayMatrix(int A[][],int len) {

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

for (int j = 0; j < len; j++)

System.out.print(A[i][j] + " ");

System.out.println();

}

}

// Perform encryption/decryption

public static void performEncryptionOrDecryption(String temp, int s)

{

while (temp.length() > s)

{

String line = temp.substring(0, s);

temp = temp.substring(s, temp.length());

findColumnMatrix(line);

if(choice ==1){

multiplyColumnByKey(line.length());

showResult(line.length());

}else if(choice==2){

multiplyColumnByInverseKey(line.length());

showResult(line.length());

}

}

if (temp.length() == s){

if(choice ==1){

findColumnMatrix(temp);

multiplyColumnByKey(temp.length());

showResult(temp.length());

}

else if(choice==2){

findColumnMatrix(temp);

multiplyColumnByInverseKey(temp.length());

showResult(temp.length());

}

}

else if (temp.length() < s)

{

for (int i = temp.length(); i < s; i++)

temp = temp + 'x';

if(choice ==1){

findColumnMatrix(temp);

multiplyColumnByKey(temp.length());

showResult(temp.length());

}

else if(choice==2){

findColumnMatrix(temp);

multiplyColumnByInverseKey(temp.length());

showResult(temp.length());

}

}

}

// Compute the key matrix

public static void findKeyMatrix(String key, int len)

{

keyMatrix = new int[len][len];

int k = 0;

for (int i = 0; i < len; i++)

{

for (int j = 0; j < len; j++)

{

keyMatrix[i][j] = ((int) key.charAt(k)) - casevariable;

k++;

}

}

if(choice==1)

{

System.out.println("\nKEY MATRIX");

System.out.println("----------");

displayMatrix(keyMatrix,len);

System.out.print("\nCipher Text : ");

}

}

// Take each group of input variables and put them into a col matrix

public static void findColumnMatrix(String line)

{

lm = new int[line.length()];

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

{

lm[i] = ((int) line.charAt(i)) - casevariable;

}

}

public static void multiplyColumnByKey(int len)

{

rm = new int[len];

for (int i = 0; i < len; i++)

{

for (int j = 0; j < len; j++)

{

rm[i] += keyMatrix[i][j] \* lm[j];

}

rm[i] %= 26;

}

}

public static void multiplyColumnByInverseKey(int len)

{

rm = new int[len];

for (int i = 0; i < len; i++)

{

for (int j = 0; j < len; j++)

{

rm[i] += inverseKeyMatrix[i][j] \* lm[j];

}

rm[i] %= 26;

}

}

public static void showResult(int len)

{

String result = "";

for (int i = 0; i < len; i++)

{

result += (char) (rm[i] + casevariable);

}

System.out.print(result);

}

public static int findDeterminant(int A[][], int N)

{

int resultOfDet;

switch (N) {

case 1:

resultOfDet = A[0][0];

break;

case 2:

resultOfDet = A[0][0] \* A[1][1] - A[1][0] \* A[0][1];

break;

default:

resultOfDet = 0;

for (int j1 = 0; j1 < N; j1++)

{

int m[][] = new int[N - 1][N - 1];

for (int i = 1; i < N; i++)

{

int j2 = 0;

for (int j = 0; j < N; j++)

{

if (j == j1)

continue;

m[i - 1][j2] = A[i][j];

j2++;

}

}

resultOfDet += Math.pow(-1.0, 1.0 + j1 + 1.0) \* A[0][j1]

\* findDeterminant(m, N - 1);

} break;

}

return resultOfDet;

}

public static void findCoFactor(int num[][], int f)

{

int b[][], fac[][];

b = new int[f][f];

fac = new int[f][f];

int p, q, m, n, i, j;

for (q = 0; q < f; q++)

{

for (p = 0; p < f; p++)

{

m = 0;

n = 0;

for (i = 0; i < f; i++)

{

for (j = 0; j < f; j++)

{

b[i][j] = 0;

if (i != q && j != p)

{

b[m][n] = num[i][j];

if (n < (f - 2))

n++;

else

{

n = 0;

m++;

}

}

}

}

fac[q][p] = (int) Math.pow(-1, q + p) \* findDeterminant(b, f - 1);

}

}

findTranspose(fac, f);

}

static void findTranspose(int fac[][], int r)

{

int i, j;

int b[][], inv[][];

b = new int[r][r];

inv = new int[r][r];

int d = findDeterminant(keyMatrix, r);

int mi = mi(d % 26);

mi %= 26;

if (mi < 0)

mi += 26;

for (i = 0; i < r; i++)

{

for (j = 0; j < r; j++)

{

b[i][j] = fac[j][i];

}

}

for (i = 0; i < r; i++)

{

for (j = 0; j < r; j++)

{

inv[i][j] = b[i][j] % 26;

if (inv[i][j] < 0)

inv[i][j] += 26;

inv[i][j] \*= mi;

inv[i][j] %= 26;

}

}

//System.out.println("\nInverse key:");

//matrixtoinverseKeyMatrixey(inv, r);

inverseKeyMatrix = inv;

if(choice==2)

{

System.out.println("\nINVERSE KEY MATRIX");

System.out.println("------------------");

displayMatrix(inverseKeyMatrix,r);

System.out.print("\nOriginal Text : ");

}

}

public static int mi(int d)

{

int q, r1, r2, r, t1, t2, t;

r1 = 26;

r2 = d;

t1 = 0;

t2 = 1;

while (r1 != 1 && r2 != 0)

{

q = r1 / r2;

r = r1 % r2;

t = t1 - (t2 \* q);

r1 = r2;

r2 = r;

t1 = t2;

t2 = t;

}

return (t1 + t2);

}

// Check if key matrix is invertible

public static boolean check(String key, int len)

{

findKeyMatrix(key, len);

int d = findDeterminant(keyMatrix, len);

d = d % 26;

if (d == 0)

{

System.out.println("Key is not invertible");

return false;

}

else if (d % 2 == 0 || d % 13 == 0)

{

System.out.println("Key is not invertible");

return false;

}

else

{

return true;

}

}

public static void main(String args[]) throws IOException

{

String key="";

BufferedReader in = new BufferedReader(new InputStreamReader(System.in));

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

System.out.println("-------");

System.out.println("1. Encrypt\n2. Decrypt\n3. Exit\n");

choice = -1;

while(choice!=3)

{

System.out.print("Enter option : ");

choice = Integer.parseInt(in.readLine());

if(choice==1)

{

System.out.print("Enter the Plain Text to Encrypt : ");

line = in.readLine();

System.out.print("Enter the Key : ");

key = in.readLine();

}

else if(choice==2)

{

System.out.print("Enter the Cipher Text to Decrypt : ");

line = in.readLine();

System.out.print("Enter the Key : ");

key = in.readLine();

}

line = line.replaceAll("\\s+","");

if(Character.isUpperCase(line.charAt(0)))

{

casevariable = 65;

}

else

{

casevariable = 97;

}

double sq = Math.sqrt(key.length());

if (sq != (long) sq)

System.out.println("Cannot Form a Square Matrix !\n");

else

{

int size = (int) sq;

if (check(key, size))

{

findCoFactor(keyMatrix, size);

performEncryptionOrDecryption(line, size);

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

}

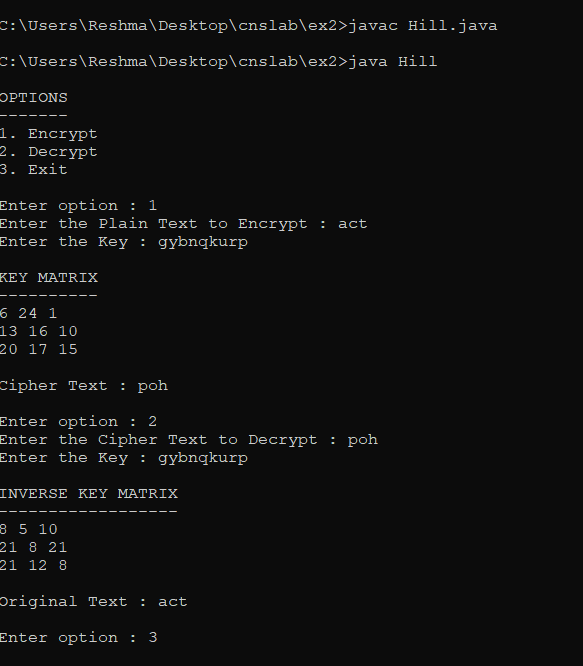
}

}

}

}

**Output:**



**Vigenere Cipher:**

**Code:**

import java.util.\*;

class VigenereCipher

{

static String generateKey(String str, String key)

{

int x = str.length();

for (int i = 0; ; i++)

{

if (x == i)

i = 0;

if (key.length() == str.length())

break;

key+=(key.charAt(i));

}

return key;

}

// returns ciphertext with the help of key

static String cipherText(String str, String key)

{

String cipher\_text = " ";

int c;

str = str.toUpperCase();

key = key.toUpperCase();

key = generateKey(str, key);

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

{

// converting in range 0-25

c = (str.charAt(i) + key.charAt(i))%26;

// convert into alphabets(ASCII)

c += 'A';

cipher\_text += (char)(c);

}

return cipher\_text;

}

// decryption

static String originalText(String cipher\_text, String keyword)

{

String orig\_text=" ";

String key = generateKey(cipher\_text, keyword);

cipher\_text = cipher\_text.toUpperCase();

key = key.toUpperCase();

for (int i = 0 ; i < cipher\_text.length() && i < key.length(); i++)

{

// converting in range 0-25

int x = (cipher\_text.charAt(i) - key.charAt(i) + 26) %26;

// convert into alphabets(ASCII)

x += 'A';

orig\_text+=(char)(x);

}

return orig\_text;

}

// Driver code

public static void main(String[] args)

{

String str, keyword, cipher\_text;

char ch;

int choice;

Scanner sc = new Scanner(System.in);

Scanner sc1 = new Scanner(System.in);

do{

System.out.println("Vigenere Cipher: \n 1. Encryption \n 2. Decryption\n");

System.out.println("Enter Choice:");

choice = sc1.nextInt();

//choice = Integer.parseInt(sc.nextLine());

switch(choice)

{

case 1:

System.out.println("Enter plain text:");

str = sc.next();

System.out.println("Enter keyword:");

keyword = sc.next();

cipher\_text = cipherText(str, keyword);

System.out.println("Ciphertext: "+cipher\_text+" \n");

break;

case 2:

System.out.println("Enter cipher text:");

cipher\_text = sc.next();

System.out.println("Enter keyword:");

keyword = sc.next();

str = originalText(cipher\_text, keyword);

System.out.println("Plain text:"+str+" \n");

break;

default: System.out.println("Invalid choice!");break;

}

System.out.println("Do you want to continue? y/n");

ch = sc.next().charAt(0);

}while(ch!='n');

sc.close();

sc1.close();

}

}

**Output:**

