**EX.NO.3 TRANSPOSITION CIPHER**

**RAIL-FENCE CIPHER – ENCRYPTION AND DECRYPTION**

**CODE:**

import java.util.Scanner;

public class RailFence {

String plainText = new String();

String cipherText = new String();

int depth;

char[][] enc\_mat;

char[][] dec\_mat;

void encrypt() {

int len = plainText.length();

boolean dir\_down = false;

int row = 0, col = 0;

enc\_mat = new char[depth][len];

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

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

enc\_mat[i][j] = '\0';

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

enc\_mat[row][col++] = plainText.charAt(i);

if (row == 0 || row == depth - 1)

dir\_down = !dir\_down;

if (dir\_down)

row++;

else

row--;

}

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

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

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

if(enc\_mat[i][j] == '\0')

System.out.print(" ");

else{

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

cipherText+=enc\_mat[i][j];

}

}

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

}

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

}

void decrypt()

{

int len = cipherText.length();

boolean dir\_down = false;

int row = 0, col = 0;

dec\_mat = new char[depth][len];

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

{

dec\_mat[row][col++] = '\*';

if (row == 0 || row == depth - 1)

dir\_down = !dir\_down;

if (dir\_down)

row++;

else

row--;

}

int index = 0;

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

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

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

if (dec\_mat[i][j] == '\*' && index<len) {

dec\_mat[i][j] = cipherText.charAt(index++);

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

}

else{

System.out.print(" ");

}

}

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

}

row = 0;

col = 0;

plainText="";

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

{

if (dec\_mat[row][col] != '\*')

plainText+=dec\_mat[row][col++];

if (row == 0 || row == depth - 1)

dir\_down = !dir\_down;

if (dir\_down)

row++;

else

row--;

}

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

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

RailFence rf = new RailFence();

System.out.println("\nRAIL FENCE CIPHER");

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

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

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

rf.plainText = sc.next();

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

rf.depth = sc.nextInt();

while (rf.depth<=0 || rf.depth>=rf.plainText.length()) {

System.out.println("\nInvalid key");

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

rf.depth = sc.nextInt();

}

rf.encrypt();

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

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

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

rf.cipherText = sc.next();

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

rf.depth = sc.nextInt();

while (rf.depth<=0 || rf.depth>=rf.plainText.length()) {

System.out.println("\nInvalid key");

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

rf.depth = sc.nextInt();

}

rf.decrypt();

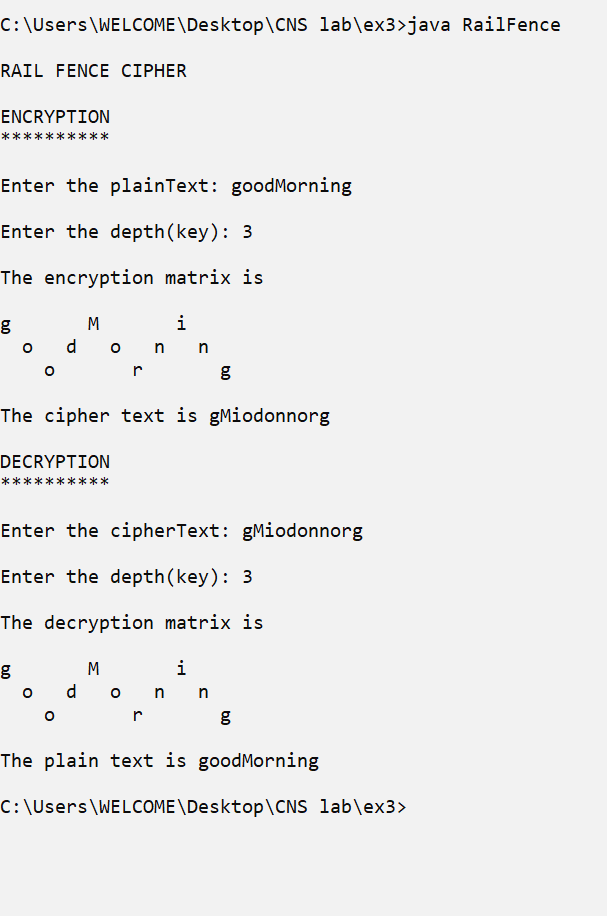
sc.close();

}

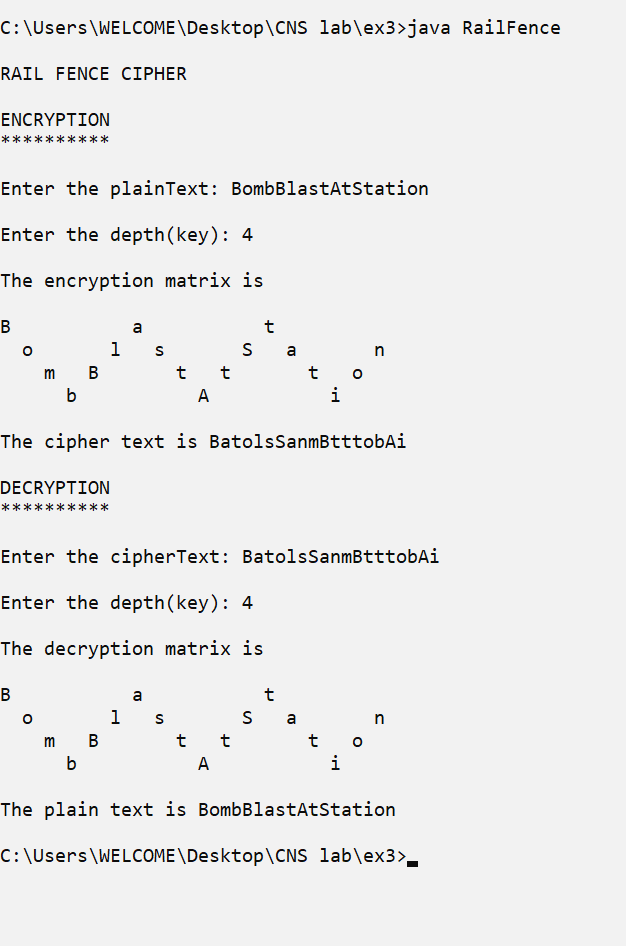
}

**OUTPUT:**

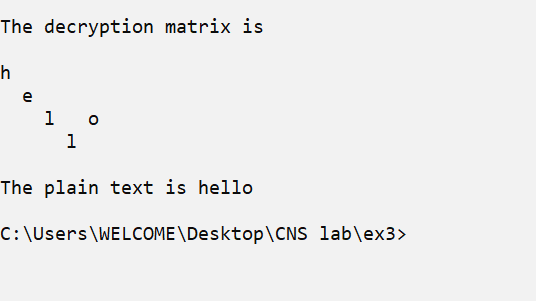
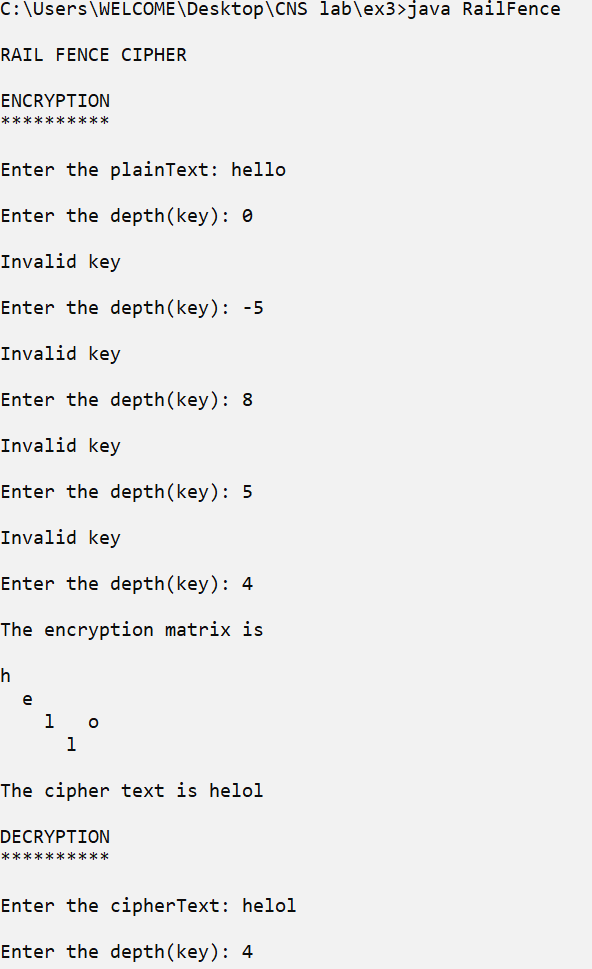
Example 1:



Example 2:



Example 3:



**ROW-COLUMN CIPHER – ENCRYPTION AND DECRYPTION**

**CODE:**

import java.util.Scanner;

public class RowColumn {

String plainText = new String();

String cipherText = new String();

String key = new String();

char[][] enc\_mat;

char[][] dec\_mat;

void encrypt() {

int n\_row, n\_col,i, j,k=0;

n\_col = key.length();

n\_row = plainText.length() / n\_col;

if (plainText.length() % n\_col>0)

n\_row += 1;

enc\_mat = new char[n\_row][n\_col];

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

for ( j = 0; j < n\_col; j++) {

System.out.print(j+1+" ");

}

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

for ( j = 0; j < n\_col; j++) {

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

}

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

for ( i = 0; i < n\_row; i++) {

for ( j = 0; j < n\_col; j++) {

if(k<plainText.length())

enc\_mat[i][j] = plainText.charAt(k++);

else

enc\_mat[i][j] = '\*';

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

}

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

}

System.out.println("\nAfter rearranging the columns using the key, the encryption matrix is \n");

for ( j = 0; j < n\_col; j++) {

System.out.print(key.charAt(j)+" ");

}

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

for ( j = 0; j < n\_col; j++) {

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

}

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

for ( i = 0; i < n\_row; i++) {

for ( j = 0; j < n\_col; j++) {

k = key.charAt(j) - 48 - 1;

System.out.print(enc\_mat[i][k] + " ");

}

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

}

for (j = 0; j < n\_col; j++) {

k = key.charAt(j) - 48 - 1;

for (i = 0; i < n\_row; i++) {

cipherText += enc\_mat[i][k];

}

}

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

}

void decrypt() {

int n\_row, n\_col,i, j,k=0;

n\_row = key.length();

n\_col = plainText.length() / n\_row;

if (plainText.length() % n\_row>0)

n\_col += 1;

dec\_mat = new char[n\_row][n\_col];

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

for ( i = 0; i < n\_row; i++) {

System.out.print(key.charAt(i) + "---> ");

for ( j = 0; j < n\_col; j++) {

dec\_mat[i][j]=cipherText.charAt(k++);

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

}

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

}

int row;

k=0;

System.out.println("\nAfter rearranging the rows using the key, the decryption matrix is \n");

for( i=0;i<cipherText.length();i=i+n\_col){

row=key.charAt(k++)-48-1;

for( j=0;j<n\_col;j++)

{

dec\_mat[row][j]=cipherText.charAt(i+j);

}

}

for ( i = 0; i < n\_row; i++) {

for ( j = 0; j < n\_col; j++) {

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

}

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

}

plainText = "";

for ( i = 0; i < n\_col; i++) {

for ( j = 0; j < n\_row; j++) {

plainText += dec\_mat[j][i];

}

}

System.out.println("\nThe plain text is " + plainText+"\n");

}

boolean validateKey(){

int len=key.length();

if(len<=1 || len>=plainText.length() || !key.matches("[1-9]+"))

return false;

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

if(!key.contains(Integer.toString(i+1)))

return false;

}

return true;

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

RowColumn rc = new RowColumn();

System.out.println("\nROW COLUMN CIPHER");

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

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

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

rc.plainText = sc.next();

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

rc.key = sc.next();

while (!rc.validateKey()) {

System.out.println("\nInvalid key");

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

rc.key = sc.next();

}

rc.encrypt();

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

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

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

rc.cipherText = sc.next();

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

rc.key = sc.next();

while (!rc.validateKey()) {

System.out.println("\nInvalid key");

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

rc.key = sc.next();

}

rc.decrypt();

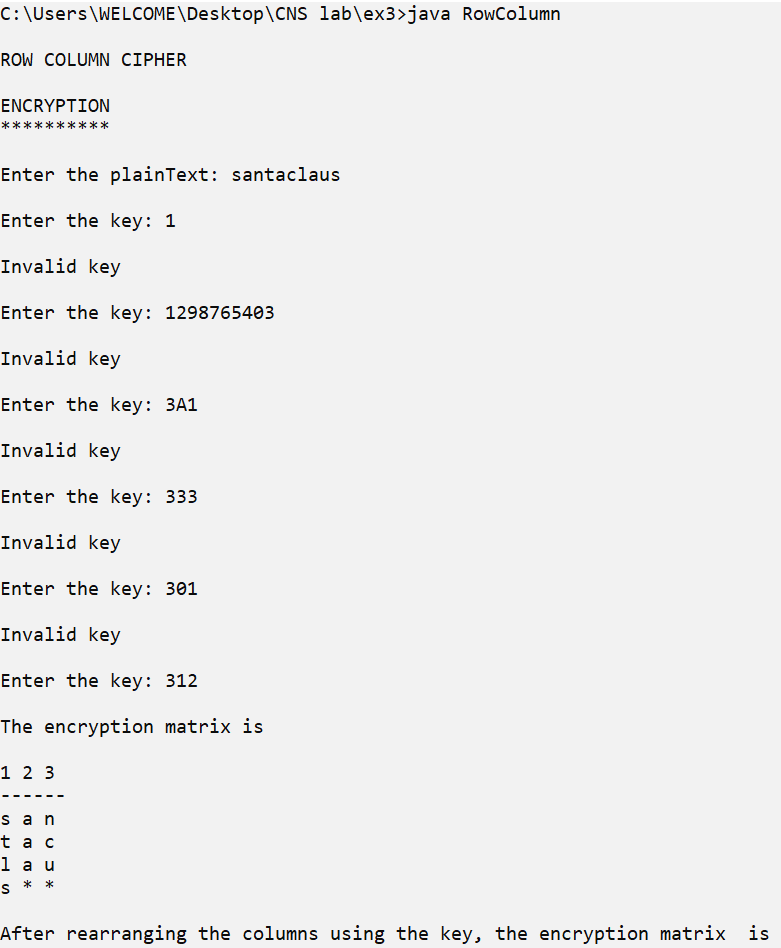
sc.close();

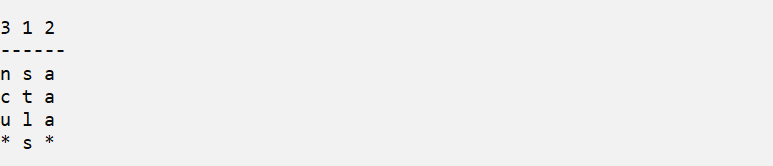
}

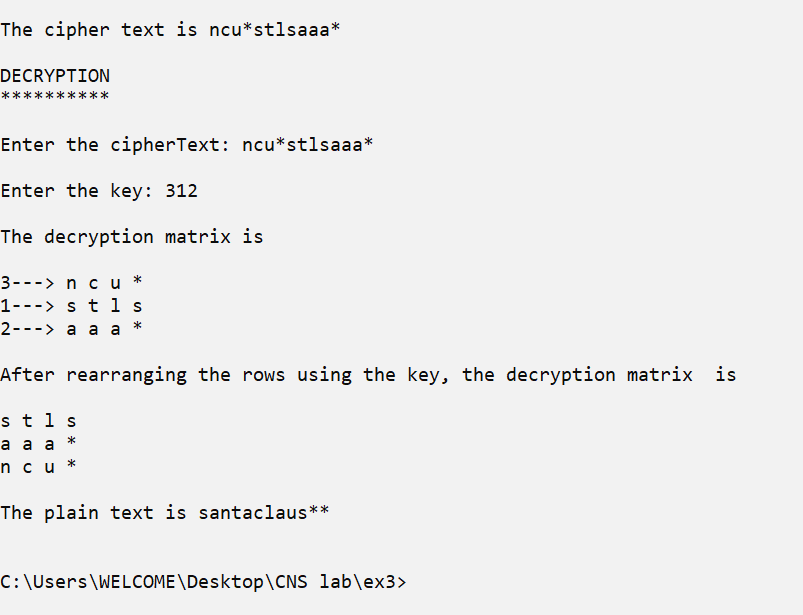
}

**OUTPUT:**

Example 1:







Example 2:

