**CODING :**

import java.util.\*;

class CaeserCipher

{

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

System.out.print("Enter the Shift value : ");

int k = input.nextInt();

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

input.nextLine();

String message = input.nextLine();

char messageArray[] = message.toLowerCase().toCharArray();

StringBuilder encryptedMessage = new StringBuilder();

for(char i : messageArray){

if(i == ' ' )

{

encryptedMessage.append(" ");

}

else{

int algo = (i+k)%123 ;

algo = algo<97 ? 97+algo : algo;

char encryption = (char) algo;

encryptedMessage.append(encryption);

// System.out.println(encryptedMessage);

}

}

System.out.println("Encrypted message : "+encryptedMessage);

//Decryption

StringBuilder decryptedMessage = new StringBuilder();

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

if(encryptedMessage.charAt(i) == ' ' )

{

decryptedMessage.append(" ");

}

else{

int algo = (encryptedMessage.charAt(i)-k) %123 ;

algo = algo<97 ? algo+26 : algo;

char decryption = (char) algo;

decryptedMessage.append(decryption);

// System.out.println(decryptedMessage);

}

}

System.out.println("Decrypted message : "+decryptedMessage);

}

}

**OUTPUT :**

Enter the Shift value : 2

Enter the message : soorya harsha

Encrypted message : uqqtac jctujc

Decrypted message : soorya harsha

Enter the Shift value : 3

Enter the message : abcdexyz

Encrypted message : defghabc

Decrypted message : abcdexyz

**CODING :**

import java.util.\*;

public class PlayfairCipher{

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

System.out.print("\n\nEnter the Key String : " );

String key = input.nextLine();

key = key.toLowerCase();

char[][] model = new char[5][5];

// Remove duplicates from key and replace j with i if exists

List<Character> list = new ArrayList<Character>();

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

if(list.contains(key.charAt(i))) continue;

else list.add(key.charAt(i));

}

System.out.println("Key after removing the duplicates : "+ list);

// Store the remaining alphabets in the 2D array

for (char i = 'a'; i<='z'; i++) {

if(list.contains(i)) {

// key contains i but not j then skip j

if(i=='i' && !list.contains('j')) i++;

// key contains j and we added i then remove j

else if(i=='j' && list.contains('i')){

list.remove('j');

}

continue;

}

else{

// key does not contain i we added i and check if j is present

// if true remove i else skip j

list.add(i);

if(i=='i' && list.contains('j')){

int a=list.indexOf('i');

list.remove(a);

}

else if(i=='i') i++;

}

}

System.out.println("Key after adding remaining alphabets : "+ list);

// Store the list in 2D array

int row,col;

row = col = 0;

for(Character c : list){

if(col<5){

model[row][col] = c;

col++;

}

else {

col=0;

row = row + 1;

model[row][col] = c;

col++;

}

}

// printing the model

for(int i=0; i<5 ;i++){

for(int j=0; j<5;j++){

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

}

System.out.println(" ");

}

// input the Message string

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

String message = input.nextLine();

message = message.toLowerCase();

int charCount =0;

// replace adjacent duplicates

StringBuilder sb = new StringBuilder();

for(int i=0; i<message.length()-1; i++){

if (message.charAt(i) == ' ') continue;

sb.append(message.charAt(i));

charCount++;

if(message.charAt(i)==message.charAt(i+1)){

sb.append("x");

charCount++;

}

//last Character logic

if(i==message.length()-2) {

sb.append(message.charAt(i+1));

charCount++;

}

}

//Even pairs checking

if(charCount % 2 != 0){

sb.append("x");

charCount++;

}

System.out.println(sb.toString());

//separating each pairs into the stringlist

List<String> stringList = new ArrayList<String>();

StringBuilder pairString = new StringBuilder();

String encryptedMessage = new String();

encryptedMessage = "";

for(int i =0;i<charCount-1;i+=2) {

pairString.append(sb.charAt(i));

pairString.append(sb.charAt(i+1));

stringList.add(pairString.toString());

pairString.setLength(0);

}

System.out.println(stringList.toString());

//Encryption algorithm

int firstRowIndex , secondRowIndex , firstColumnIndex , secondColumnIndex;

firstRowIndex = firstColumnIndex = secondColumnIndex = secondRowIndex = -1;

for(String pair : stringList){

//retrive the row and column number to check whether the pair occurs in the same row or column

for(int i = 0; i < 5; i++){

for(int j = 0; j < 5; j++){

if(model[i][j] == pair.charAt(0)) {

firstRowIndex = i;

firstColumnIndex = j;

}

if(model[i][j] == pair.charAt(1)){

secondRowIndex = i;

secondColumnIndex = j;

}

}

}

// same row logic

if(firstRowIndex == secondRowIndex) {

//first letter

int shiftIndex = (firstColumnIndex + 1)%5;

encryptedMessage = encryptedMessage+ model[firstRowIndex][ shiftIndex ];

//second letter

shiftIndex = (secondColumnIndex + 1)%5;

encryptedMessage = encryptedMessage+ model[secondRowIndex][ shiftIndex ];

}

//same column logic

else if(firstColumnIndex == secondColumnIndex){

//first

int shiftIndex = (firstRowIndex + 1)%5;

encryptedMessage = encryptedMessage+ model[ shiftIndex ][firstColumnIndex];

//second

shiftIndex = (secondRowIndex + 1)%5;

encryptedMessage = encryptedMessage+ model[ shiftIndex ][secondColumnIndex];

}

//Neither same row nor same column logic

else{

//first letter

encryptedMessage = encryptedMessage+ model[firstRowIndex][ secondColumnIndex];

encryptedMessage = encryptedMessage+ model[secondRowIndex][ firstColumnIndex];

}

System.out.println("The Encrypted Message : " + encryptedMessage );

}

System.out.println("The Encrypted Message : " + encryptedMessage );

//Decryption algorithm

List<String> encryptedMessageList = new ArrayList<String>();

StringBuilder encryptedPairString = new StringBuilder();

String decryptedMessage = new String();

decryptedMessage = "";

// Making the encrypted message pairs

for(int i =0;i<charCount-1;i+=2) {

pairString.append(encryptedMessage.charAt(i));

pairString.append(encryptedMessage.charAt(i+1));

encryptedMessageList.add(pairString.toString());

pairString.setLength(0);

}

System.out.println(encryptedMessageList.toString());

for(String pair : encryptedMessageList){

//retrive the row and column number to check whether the pair occurs in the same row or column

for(int i = 0; i < 5; i++){

for(int j = 0; j < 5; j++){

if(model[i][j] == pair.charAt(0)) {

firstRowIndex = i;

firstColumnIndex = j;

}

if(model[i][j] == pair.charAt(1)){

secondRowIndex = i;

secondColumnIndex = j;

}

}

}

// same row logic

if(firstRowIndex == secondRowIndex) {

//first letter

int shiftIndex = (firstColumnIndex - 1+5)%5;

decryptedMessage += model[firstRowIndex][ shiftIndex ];

//second letter

shiftIndex = (secondColumnIndex - 1 +5)%5;

decryptedMessage += model[secondRowIndex][ shiftIndex ];

}

//same column logic

else if(firstColumnIndex == secondColumnIndex){

//first

int shiftIndex = (firstRowIndex - 1+5)%5;

decryptedMessage += model[ shiftIndex ][firstColumnIndex];

//second

shiftIndex = (secondRowIndex - 1+5)%5;

decryptedMessage += model[ shiftIndex ][secondColumnIndex];

}

//Neither same row nor same column logic

else{

decryptedMessage = decryptedMessage + model[firstRowIndex][ secondColumnIndex];

decryptedMessage = decryptedMessage + model[secondRowIndex][ firstColumnIndex];

}

System.out.println("The Decrypted Message : " + decryptedMessage );

}

System.out.println("The Decrypted Message : " + decryptedMessage );

}

}

**OUTPUT :**

Enter the Key String : monarchy

Key after removing the duplicates : [m, o, n, a, r, c, h, y]

Key after adding remaining alphabets : [m, o, n, a, r, c, h, y, b, d, e, f, g, i, k, l, p, q, s, t, u, v, w, x, z]

m o n a r

c h y b d

e f g i k

l p q s t

u v w x z

Enter the Message string : lapptop

lapxptop

[la, px, pt, op]

The Encrypted Message : sm

The Encrypted Message : smsv

The Encrypted Message : smsvql

The Encrypted Message : smsvqlhv

The Encrypted Message : smsvqlhv

[sm, sv, ql, hv]

The Decrypted Message : la

The Decrypted Message : lapx

The Decrypted Message : lapxpt

The Decrypted Message : lapxptop

The Decrypted Message : lapxptop

**CODING :**

import java.util.\*;

public class HillCipher {

static int flag = 0;

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

String plainText;

int n;

System.out.print("Enter the N value : " );

n = input.nextInt();

int[][] model = new int[n][n];

System.out.print("Enter the key values :");

// Create a model using the key

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

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

model[i][j] = input.nextInt();

//System.out.println("model[i][j]: " + model[i][j]);

}

}

System.out.println();

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

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

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

}

System.out.println();

}

input.nextLine();

//Split the plain text into n characters

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

plainText = input.nextLine();

List<Integer> threeLetters = new ArrayList<Integer>();

StringBuilder encryptedMessage = new StringBuilder();

StringBuilder encryptedThreeLetters = new StringBuilder();

int s=0;

while(true){

threeLetters = splitIntoThree(threeLetters,plainText.substring(s, plainText.length()) );

// space logic

if(flag==0) s=s+n;

else if (flag==1) s=s+n+1;

System.out.println(threeLetters);

//Encrption

encryptedThreeLetters = encryption(threeLetters,model,n);

encryptedMessage.append(encryptedThreeLetters);

System.out.println("Encrypted message: " + encryptedMessage);

if(s >= plainText.length() ) break;

}

//Decryption

//Inverse of the key matrix

float[][] inverseModel = new float[n][n];

//inverseModel = inverseMatrix(model,n);

// }

}

static List<Integer> splitIntoThree(List<Integer> threeLetters, String text){

flag =0;

threeLetters.clear();

int count = 0;

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

if(text.charAt(i)==' ') {

flag =1;

continue;

}

else{

threeLetters.add(text.charAt(i)-97);

//System.out.println(threeLetters);

count++;

}

if(count == 3) break;

}

return threeLetters;

}

static StringBuilder encryption(List<Integer> threeLetters,int[][] model,int n){

StringBuilder sb = new StringBuilder();

sb.setLength(0);

int sum;

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

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

sum=0;

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

sum += model[k][j] \* threeLetters.get(k);

System.out.println("Sum : "+sum);

}

sum = (sum % 26)+ 97;

System.out.println("Sum after mod and +a : "+sum);

sb.append((char)sum);

System.out.println("Equivalent words : "+sb);

}

}

return sb;

}

}

**OUTPUT :**

Enter the N value : 3

Enter the key values :1 2 3 4 5 6 7 8 9

1 2 3

4 5 6

7 8 9

Enter the message : helloo

[7, 4, 11]

Sum : 7

Sum : 23

Sum : 100

Sum after mod and +a : 119

Equivalent words : w

Sum : 14

Sum : 34

Sum : 122

Sum after mod and +a : 115

Equivalent words : ws

Sum : 21

Sum : 45

Sum : 144

Sum after mod and +a : 111

Equivalent words : wso

Encrypted message: wso

[11, 14, 14]

Sum : 11

Sum : 67

Sum : 165

Sum after mod and +a : 106

Equivalent words : j

Sum : 22

Sum : 92

Sum : 204

Sum after mod and +a : 119

Equivalent words : jw

Sum : 33

Sum : 117

Sum : 243

Sum after mod and +a : 106

Equivalent words : jwj

Encrypted message: wsojwj

**CODING :**

import java.util.\*;

public class vigeners {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

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

String key = input.nextLine();

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

String plainText = input.nextLine();

String encryptedMessage = EncryptionAlgo(key, plainText);

System.out.println();

String decryptedMessage = DecryptionAlgo(key, encryptedMessage);

}

static String EncryptionAlgo(String key, String plainText){

String encryptedMessage = "";

int loopCount = plainText.length();

int i=0;

int j=0;

while(loopCount>0) {

char c = (char) ((((plainText.charAt(i) - 'a') + (key.charAt(j) - 'a') ) % 26) + 'a') ;

encryptedMessage += c;

System.out.println("Encrypted Message : "+encryptedMessage);

i++;

j= (j+1)%key.length();

loopCount--;

}

return encryptedMessage;

}

static String DecryptionAlgo(String key, String encryptedMessage){

String decryptedMessage = "";

int loopCount = encryptedMessage.length();

int i=0;

int j=0;

while(loopCount>0) {

char c = (char) ((((encryptedMessage.charAt(i) - 'a') - (key.charAt(j) - 'a') +26 ) % 26) + 'a') ;

decryptedMessage += c;

System.out.println("Decrypted Message : "+decryptedMessage);

i++;

j= (j+1)%key.length();

loopCount--;

}

return decryptedMessage;

}

}

**OUTPUT :**

Enter the key : security

Enter the Plain Text : meetmeatthepark

Encrypted Message : eigndmtrllgjrzd

Decrypted Message : meetmeatthepark

**CODING :**

import java.util.\*;

class ColumnTransposition{

public static void display(char[][] arr){

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

for(int j=0;j<arr[i].length;j++){

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

}

System.out.println();

}

}

public static String encrypt(String plaintext, String key) {

int keyLength = key.length();

int numRows = (int) Math.ceil((double) plaintext.length() / keyLength);

char[][] grid = new char[numRows][keyLength];

// Fill the grid with the plaintext

int plaintextIndex = 0;

for (int row = 0; row < numRows; row++) {

for (int col = 0; col < keyLength; col++) {

if (plaintextIndex < plaintext.length()) {

// if(plaintext.charAt(plaintextIndex) == ' ') {

// plaintextIndex++;

// continue;

// }

grid[row][col] = plaintext.charAt(plaintextIndex);

plaintextIndex++;

} else {

grid[row][col] = ' ';

}

}

}

display(grid);

// Create an array to store the column order

int[] order = new int[keyLength];

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

order[i] = key.charAt(i);

}

// Sort the order array and apply the same permutation to the grid

Arrays.sort(order);

StringBuilder ciphertext = new StringBuilder();

char[][] encryptedGrid = new char[numRows][keyLength];

for (int col = 0; col < keyLength; col++) {

int originalIndex = key.indexOf(order[col]);

for (int row = 0; row < numRows; row++) {

ciphertext.append(grid[row][originalIndex]);

}

}

// System.out.println("EncryptedGrid: " );

// display(encryptedGrid);

return ciphertext.toString();

}

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

System.out.print("Enter the Plain text : ");

String plaintext = input.nextLine();

System.out.print("Enter the key String : ");

String key = input.nextLine();

String ciphertext = encrypt(plaintext, key);

System.out.println("Ciphertext: " + ciphertext);

}

}

**OUTPUT :**

Enter the Plain text : CRYTOGRAPHY

Enter the key String : XABCD

C R Y T O

G R A P H

Y

Ciphertext: RR YA TP OH CGY