**Practical no - 1 Roll no-17,19**

1. **Caecar cipher**

import java.io.\*;

import java.util.\*;

class caesercipher1{

public static StringBuffer encrypt(String text,int s){

StringBuffer sb=new StringBuffer();

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

if(Character.isUpperCase(text.charAt(i))){

char ch=(char)(((int)text.charAt(i)+s-65)%26+65);

sb.append(ch);

}

else if(Character.isLowerCase(text.charAt(i))){

char ch=(char)(((int)text.charAt(i)+s-97)%26+97);

sb.append(ch);

}

else{

System.out.println("Error");

}

}

return sb;

}

public static void main(String args[]) throws Exception{

InputStreamReader is=new InputStreamReader(System.in);

BufferedReader sc=new BufferedReader(is);

System.out.println("Enter the text");

String text=sc.readLine();

System.out.println("Enter the key");

int s=Integer.parseInt(sc.readLine());

System.out.println("Cipher is " + encrypt(text,s));

}

}

**Output:**

C:\Users\Ashish\Desktop\programs\tycs\ins>javac caesercipher1.java

C:\Users\Ashish\Desktop\programs\tycs\ins>java caesercipher1

Enter the text

helloworld

Enter the key

3

Cipher is khoorzruog

**Roll no-17,19**

1. **Monoalphabetic cipher**

import java.io.\*;

import java.util.\*;

class monoalphabetic{

public static String doEncrypt(String PT,String key)

{

int idx;

char c;

StringBuffer sb=new StringBuffer(PT);

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

idx=sb.charAt(i)-65;

c=key.charAt(idx);

sb.setCharAt(i,c);

}

return new String(sb);

}

public static String doDecrypt(String PT,String key)

{

int idx;

char c;

StringBuffer sb=new StringBuffer(PT);

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

c=sb.charAt(i);

idx=getIndex(c,key);

c=(char)(idx+65);

sb.setCharAt(i,c);

}

return new String(sb);

}

public static int getIndex(char c,String key){

int idx=0;

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

if(key.charAt(i)==c){

idx=i;

}

}

return idx;

}

public static void main(String args[]) throws Exception{

InputStreamReader is=new InputStreamReader(System.in);

BufferedReader sc=new BufferedReader(is);

System.out.println("Enter the text");

String text=sc.readLine().toUpperCase();

String key="QWERTYUIOPASDFGHJKLZXCVBNM";

System.out.println("Cipher is " + doEncrypt(text,key));

System.out.println("After Decrytion : "+doDecrypt(doEncrypt(text,key),key));

}

}

**Output:**

C:\Users\Ashish\Desktop\programs\tycs\ins>javac monoalphabetic.java

C:\Users\Ashish\Desktop\programs\tycs\ins>java monoalphabetic

Enter the text

helloworld

Cipher is ITSSGVGKSR

After Decrytion : HELLOWORLD

**Practical no - 2 Roll no-17,19**

1. **Vernam cipher**

import java.io.\*;

class Vernam

{

public static int getCharValue(char x)

{

int y=(int)'a';

return ((int)x-y);

}

public static char getNumberValue(int x)

{

int z=x+(int)'a';

return ((char)z);

}

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

{

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

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

String accept=br.readLine();

System.out.println("\nEnter your one time pad text");

String pad=br.readLine();

int aval[]=new int[accept.length()];

int pval[]=new int[pad.length()];

int initval[]=new int[pad.length()];

if(pad.length()!=accept.length())

{

System.out.println("Invalid one time pad. Application terminates.");

return;

}

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

{

int k=getCharValue(accept.charAt(i));

aval[i]=k;

}

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

{

int k=getCharValue(pad.charAt(i));

pval[i]=k;

}

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

{

initval[i]=aval[i]+pval[i];

if(initval[i]>25)

initval[i]-=26;

}

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

String cipher="";

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

{

cipher+=getNumberValue(initval[i]);

}

System.out.print(cipher);

}

}

**Output :**

C:\Users\Ashish\Desktop\programs\tycs\ins>java Vernam

Enter your plain text

helloworld

Enter your one time pad text

asdfghjhka

Cipher text is :

Hwoqudxyvd

1. **Playfair Cipher Rollno-17,19**

import java.util.Scanner;

class Basic{

String allChar="ABCDEFGHIJKLMNOPQRSTUVWXYZ";

boolean indexOfChar(char c)

{

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

{

if(allChar.charAt(i)==c)

return true;

}

return false;

}

}+

class PlayFair{

Basic b=new Basic();

char keyMatrix[][]=new char[5][5];

boolean repeat(char c)

{

if(!b.indexOfChar(c))

{

return true;

}

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

{

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

{

if(keyMatrix[i][j]==c || c=='J')

return true;

}

}

return false;

}

void insertKey(String key)

{

key=key.toUpperCase();

key=key.replaceAll("J", "I");

key=key.replaceAll(" ", "");

int a=0,b=0;

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

{

if(!repeat(key.charAt(k)))

{

keyMatrix[a][b++]=key.charAt(k);

if(b>4)

{

b=0;

a++;

}

}

}

char p='A';

while(a < 5)

{

while(b < 5)

{

if(!repeat(p))

{

keyMatrix[a][b++]=p;

}

p++;

}

b=0;

a++;

}

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

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

{

System.out.println();

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

{

System.out.print("\t"+keyMatrix[i][j]);

}

}

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

}

int rowPos(char c)

{

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

{

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

{

if(keyMatrix[i][j]==c)

return i;

}

}

return -1;

}

int columnPos(char c)

**Rollno-17,19**

{

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

{

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

{

if(keyMatrix[i][j]==c)

return j;

}

}

return -1;

}

String encryptChar(String plain)

{

plain=plain.toUpperCase();

char a=plain.charAt(0),b=plain.charAt(1);

String cipherChar="";

int r1,c1,r2,c2;

r1=rowPos(a);

c1=columnPos(a);

r2=rowPos(b);

c2=columnPos(b);

if(c1==c2)

{

++r1;

++r2;

if(r1>4)

r1=0;

if(r2>4)

r2=0;

cipherChar+=keyMatrix[r1][c2];

cipherChar+=keyMatrix[r2][c1];

}

else if(r1==r2)

{

++c1;

++c2;

if(c1>4)

c1=0;

if(c2>4)

c2=0;

cipherChar+=keyMatrix[r1][c1];

cipherChar+=keyMatrix[r2][c2];

}

else{

cipherChar+=keyMatrix[r1][c2];

cipherChar+=keyMatrix[r2][c1];

}

return cipherChar;

}

String Encrypt(String plainText,String key)

{

insertKey(key);

String cipherText="";

plainText=plainText.replaceAll("j", "i");

plainText=plainText.replaceAll(" ", "");

plainText=plainText.toUpperCase();

int len=plainText.length();

// System.out.println(plainText.substring(1,2+1));

if(len/2!=0)

{

plainText+="X";

++len;

}

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

{

cipherText+=encryptChar(plainText.substring(i,i+2));

cipherText+=" ";

}

return cipherText;

}

String decryptChar(String cipher)

{

cipher=cipher.toUpperCase();

char a=cipher.charAt(0),b=cipher.charAt(1);

String plainChar="";

int r1,c1,r2,c2;

r1=rowPos(a);

c1=columnPos(a);

r2=rowPos(b);

c2=columnPos(b);

if(c1==c2)

{

**Rollno-17,19**

--r1;

--r2;

if(r1 < 0)

r1=4;

if(r2 < 0)

r2=4;

plainChar+=keyMatrix[r1][c2];

plainChar+=keyMatrix[r2][c1];

}

else if(r1==r2)

{

--c1;

--c2;

if(c1 < 0)

c1=4;

if(c2 < 0)

c2=4;

plainChar+=keyMatrix[r1][c1];

plainChar+=keyMatrix[r2][c2];

}

else{

plainChar+=keyMatrix[r1][c2];

plainChar+=keyMatrix[r2][c1];

}

return plainChar;

}

String Decrypt(String cipherText,String key)

{

String plainText="";

cipherText=cipherText.replaceAll("j", "i");

cipherText=cipherText.replaceAll(" ", "");

cipherText=cipherText.toUpperCase();

int len=cipherText.length();

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

{ plainText+=decryptChar(cipherText.substring(i,i+2));

plainText+=" ";

}

return plainText;

}

}

class PlayFairCipher2{

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

{

PlayFair p=new PlayFair();

Scanner scn=new Scanner(System.in);

String key,cipherText,plainText;

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

plainText=scn.nextLine();

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

key=scn.nextLine();

cipherText=p.Encrypt(plainText,key);

System.out.println("Encrypted text:");

System.out.println("---------------------------------------------------------\n"+cipherText);

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

String encryptedText=p.Decrypt(cipherText, key);

System.out.println("Decrypted text:" );

System.out.println("---------------------------------------------------------\n"+encryptedText);

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

}

}

**Output :**

E:\laxmi>javac PlayFairCipher2.java

E:\laxmi>java PlayFairCipher2

Enter plaintext:

network

Enter Key:

cipher

-------------------------Key Matrix-------------------

        C       I       P       H       E

        R       A       B       D       F

        G       K       L       M       N

        O       Q       S       T       U

        V       W       X       Y       Z

---------------------------------------------------------

Encrypted text:

---------------------------------------------------------

UF QY VG LW

---------------------------------------------------------

Decrypted text:

---------------------------------------------------------

NE TW OR KX

**Practical no - 3 Roll no-17,19**

1. **Railfence Cipher**

import java.util.\*;

import java.io.\*;

class RailfenceDynamic

{

public static void main(String [] args)

{

Scanner sc=new Scanner(System.in);

System.out.println("1.Encrypt \n 2.Decryption");

int choice=sc.nextInt();

if(choice==1)

{

System.out.println("Encryption");

cipherEncryption();

}

else if(choice==2)

{

System.out.println("Decryption");

}

else

{

System.out.println("Incorrect choice");

}

}

public static void cipherEncryption()

{

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

Scanner sc=new Scanner(System.in);

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

String message=sc.next();

message+=sc.next();

int l=message.length();

//removing white space from message

message=message.replaceAll("\\s","");

sc.nextLine();

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

int key=sc.nextInt();

char[][]rail=new char[key][l];

//matrix

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

{

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

{

rail[i][j]='\_';

}

}

int row=0;

int check=0;

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

{

if(check==0)

{

rail[row][i]=message.charAt(i);

row++;

if(row==key)

{

check=1;

row--;

}

}

else if(check==1)

{

row--;

rail[row][i]=message.charAt(i);

if(row==0)

{

check=0;

row=1;

}

}

}

String encrypText="";

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

{

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

{

encrypText+=rail[i][j];

System.out.print(rail[i][j]);

}

System.out.println();

}

encrypText=encrypText.replaceAll("\_","");

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

}

}

**Output :**

C:\Users\Ashish\Desktop\programs\tycs\ins>java RailfenceDynamic

1.Encrypt

2.Decryption

1

Encryption

Enter Message:hellowworldhello

Enter Key:3

h\_\_\_o\_\_\_r\_\_\_e\_\_\_

\_e\_l\_w\_o\_l\_h\_l\_o

\_\_l\_\_\_w\_\_\_d\_\_\_l\_

Encrypted message:horeelwolhlolwdl

**Roll no-17,19**

1. **Simple Columnar Technique**

import java.io.\*;

import java.util.\*;

class SCT{

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

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

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

String accept=br.readLine();

System.out.println("Enter the no of rows ");

int r=Integer.parseInt(br.readLine());

System.out.println("Enter the no of cols");

int c=Integer.parseInt(br.readLine());

char[][] ch=new char[r][c];

int count=0;

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

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

if(count>=accept.length()){

ch[i][j]=' ';

count++;

}

else{

ch[i][j]=accept.charAt(count);

count++;

}

}

}

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

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

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

}

System.out.println(" ");

}

System.out.println("\nEnter the order of cols you want to view them in");

int choice[]=new int[c];

for(int k=0;k<c;k++)

{

System.out.println("Choice "+k+"-> ");

choice[k]=Integer.parseInt(br.readLine());

}

System.out.println("\nCipher text in matrix is ->");

String cipher="";

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

{

int k=choice[j];

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

{

cipher+=ch[i][k];

}

}

cipher=cipher.trim();

System.out.println(cipher);

}

}

**Output:**

C:\Users\Ashish\Desktop\programs\tycs\ins>java SCT

Enter your plain text

helloworld

Enter the no of rows

3

Enter the no of cols

3

h e l

l o w

o r l

Enter the order of cols you want to view them in

Choice 0->

2

Choice 1->

1

Choice 2->

0

Cipher text in matrix is ->

lwleorhlo