# **Transposition Cipher**

import java.util.\*;

import java.util.Scanner; // needed for Scanner

public class transpositionCipher

{

public static void main(String args[])

{

String key;

String message;

String encryptedMessage;

// Letters in the x-axis

int x=0;

// Letters in the y-axis

int y=0;

// Prompt the user

System.out.print( "Type your Key : " );

// Read a line of text from the user.

Scanner scan = new Scanner(System.in);

key = scan.nextLine();

// Display the input back to the user.

System.out.println( "Your Key is " + key );

//Prompt the user

System.out.print( "Type your Message : " );

//Read a line of text from the user.

message = scan.nextLine();

//Display the input back to the user.

System.out.println( "Your Message is " + message );

int msgchar = message.length();

int keycahr = key.length();

if (!((msgchar % keycahr) == 0)){

do{

message = message + "x";

msgchar = message.length();

}while(!((msgchar % keycahr) == 0));

}

encryptedMessage = "";

// To set the temp as [x][y]

char temp[][]=new char [key.length()][message.length()];

char msg[] = message.toCharArray();

// To populate the array

x=0;

y=0;

// To convert the message into an array of char

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

{

temp[x][y]=msg[i];

if (x==(key.length()-1))

{

x=0;

y=y+1;

} // Close if

else

{

x++;

}

} // Close for loop

// To sort the key

char t[]=new char [key.length()];

t=key.toCharArray();

Arrays.sort(t);

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

{

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

{

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

}

System.out.println();

}

System.out.println();

// To print out row by row (i.e. y)

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

// To compare the the sorted Key with the key

// For char in the key

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

int pos=0;

// To get the position of key.charAt(i) from sorted key

for (pos=0;pos<t.length;pos++){

if (key.charAt(i)==t[pos]){

// To break the for loop once the key is found

break;

}

}

System.out.print(temp[pos][j]);

encryptedMessage+=temp[pos][j];

}

System.out.println();

}

System.out.println(encryptedMessage);

System.exit(0);

}

}

# **DES**

//Java Programme

package descrypt;

import java.util.\*;

import javax.crypto.\*;

import java.io.\*;

import java.security.InvalidKeyException;

import java.security.NoSuchAlgorithmException;

import java.security.spec.InvalidKeySpecException;

public class descrypt {

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

NoSuchAlgorithmException, InvalidKeyException, InvalidKeySpecException,

NoSuchPaddingException, IllegalBlockSizeException, BadPaddingException {

try {

BufferedReader bfn=new BufferedReader(new

InputStreamReader(System.in));

System.out.println("Enter the String: ");

String str=bfn.readLine();

byte[] msg=str.getBytes();

KeyGenerator Mygenerator=KeyGenerator.getInstance("DES");

SecretKey mydeskey=Mygenerator.generateKey();

Cipher myCipher = Cipher.getInstance("DES");

myCipher.init(Cipher.ENCRYPT\_MODE, mydeskey);

byte[] myEncryptedBytes=myCipher.doFinal(msg);

myCipher.init(Cipher.DECRYPT\_MODE, mydeskey);

byte[] myDecryptedBytes=myCipher.doFinal(myEncryptedBytes);

String encrypteddata=new String(myEncryptedBytes);

String decrypteddata=new String(myDecryptedBytes);

System.out.println("Encrypted - "+ encrypteddata);

System.out.println("Decrypted Message - "+ decrypteddata);

}

catch(Exception e) {

System.out.println("Exception caught!!!");

}}}

# **AES**

import javax.crypto.\*;

import javax.crypto.spec.IvParameterSpec;

import java.security.InvalidAlgorithmParameterException;

import java.security.InvalidKeyException;

import java.security.NoSuchAlgorithmException;

import java.util.Base64;

import java.io.\*;

public class AESCrypt{

public static void main(String args[]) throws

NoSuchAlgorithmException,NoSuchPaddingException,

InvalidKeyException, IllegalBlockSizeException, BadPaddingException,

InvalidAlgorithmParameterException,IOException {

//KeyGeneration

KeyGenerator keygenerator=KeyGenerator.getInstance("AES");

SecretKey secretkey=keygenerator.generateKey();

String

secretKeyString=Base64.getEncoder().encodeToString(secretkey.getEncoded());

System.out.println("generated key: "+secretKeyString);

//Encryption of message

Cipher encryptionCipher=Cipher.getInstance("AES/CBC/PKCS5Padding");

byte[] InitVectorBytes=keygenerator.generateKey().getEncoded();

IvParameterSpec parameterSpec=new IvParameterSpec(InitVectorBytes);

encryptionCipher.init(Cipher.ENCRYPT\_MODE,secretkey,parameterSpec);

BufferedReader bfn=new BufferedReader(new

InputStreamReader(System.in));

System.out.println("Enter the String: ");

String str=bfn.readLine();

byte[] encryptedMessageBytes=encryptionCipher.doFinal(str.getBytes());

String

encryptedMessage=Base64.getEncoder().encodeToString(encryptedMessageBytes);

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

//Decryption of message

Cipher decryptioncipher=Cipher.getInstance("AES/CBC/PKCS5Padding");

decryptioncipher.init(Cipher.DECRYPT\_MODE,secretkey,parameterSpec);

byte[]

decryptedMessageBytes=decryptioncipher.doFinal(encryptedMessageBytes);

String decryptedMessage=new String(decryptedMessageBytes);

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

}}

# **RSA**

// Java Program to Implement the RSA Algorithm

import java.math.\*;

import java.util.\*;

class RSA {

public static void main(String args[])

{

int p, q, n, z, d = 0, e, i;

// The number to be encrypted and decrypted

int msg = 12;

double c;

BigInteger msgback;

// 1st prime number p

p = 3;

// 2nd prime number q

q = 11;

n = p \* q;

z = (p - 1) \* (q - 1);

System.out.println("the value of z = " + z);

for (e = 2; e < z; e++) {

// e is for public key exponent

if (gcd(e, z) == 1) {

break;

}

}

System.out.println("the value of e = " + e);

for (i = 0; i <= 9; i++) {

int x = 1 + (i \* z);

// d is for private key exponent

if (x % e == 0) {

d = x / e;

break;

}

}

System.out.println("the value of d = " + d);

c = (Math.pow(msg, e)) % n;

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

// converting int value of n to BigInteger

BigInteger N = BigInteger.valueOf(n);

// converting float value of c to BigInteger

BigInteger C = BigDecimal.valueOf(c).toBigInteger();

msgback = (C.pow(d)).mod(N);

System.out.println("Decrypted message is : "

+ msgback);

}

static int gcd(int e, int z)

{

if (e == 0)

return z;

else

return gcd(z % e, e);

}

}

# **MD5**

import java.math.BigInteger;

import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException;

import java.util.\*;

import java.io.\*;

// Java program to calculate MD5 hash value

public class MD5 {

public static String getMd5(String input)

{

try {

// Static getInstance method is called with hashing MD5

MessageDigest md = MessageDigest.getInstance("MD5");

// digest() method is called to calculate message digest

// of an input digest() return array of byte

byte[] messageDigest = md.digest(input.getBytes());

// Convert byte array into signum representation

BigInteger no = new BigInteger(1, messageDigest);

// Convert message digest into hex value

String hashtext = no.toString(16);

while (hashtext.length() < 32) {

hashtext = "0" + hashtext;

}

return hashtext;

}

// For specifying wrong message digest algorithms

catch (NoSuchAlgorithmException e) {

throw new RuntimeException(e);

}

}

// Driver code

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

{

System.out.println("Enter Text ");

Scanner sc=new Scanner(System.in);

String s= sc.next();

System.out.println("Your HashCode Generated by MD5 is: " + getMd5(s));

}

}