1. **Write a C program that contains a string (char pointer) with a value ‘Hello world’. The program**

**should XOR each character in this string with 0 and displays the result.**

#include <stdio.h>

#include <string.h>

int main()

{

char str[] = "Hello world";

for(int i=0;i<strlen(str);i++)

{

str[i] = str[i]^0;

}

printf("String after XOR with 0:%s",str);

return 0;

}

2. **Write a C program that contains a string (char pointer) with a value ‘Hello world’. The program**

**should AND or and XOR each character in this string with 127 and display the result.**

#include <stdio.h>

#include <string.h>

int main()

{

char str[] = "Hello world";

for(int i=0;i<strlen(str);i++)

{

str[i] = str[i]&127;

}

printf("String after AND with 127:%s",str);

char str1[] = "Hello World";

for(int i=0;i<strlen(str1);i++)

str1[i] = str1[i]^127;

printf("\nString after XOR with 127:%s",str1);

char str2[] = "Hello World";

for(int i=0;i<strlen(str2);i++)

str2[i] = str2[i]|127;

printf("\nString after OR with 127:%s",str2);

return 0;

}

**3. Write a Java program to perform encryption and decryption using the following algorithms**

**a. Ceaser cipher**

import java.util.\*;

public class CaesarCipher {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

String plaintext = scanner.next();

int key = 3;

// Encryption

String cipherText = new String();

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

char ch = plaintext.charAt(i);

if (Character.isUpperCase(ch)) {

ch = (char) (((ch - 'A' + key) % 26) + 'A');

} else {

ch = (char) (((ch - 'a' + key) % 26) + 'a');

}

cipherText+=ch;

}

System.out.println("Cipher text: " + cipherText);

// Decryption

String decryptedText = new String();

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

char ch = cipherText.charAt(i);

if (Character.isUpperCase(ch)) {

ch = (char) (((ch - 'A' - key + 26) % 26) + 'A');

} else {

ch = (char) (((ch - 'a' - key + 26) % 26) + 'a');

}

decryptedText+=ch;

}

System.out.println("\nDecrypted Text: " + decryptedText);

scanner.close();

}

}

**b. Substitution cipher**

import java.util.\*;

public class SubstitutionCipher {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

String plaintext = scanner.next();

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

int key = scanner.nextInt();

// Encryption

String cipherText = new String();

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

char ch = plaintext.charAt(i);

if (Character.isUpperCase(ch)) {

ch = (char) (((ch - 'A' + key) % 26) + 'A');

} else {

ch = (char) (((ch - 'a' + key) % 26) + 'a');

}

cipherText+=ch;

}

System.out.println("Cipher text: " + cipherText);

// Decryption

String decryptedText = new String();

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

char ch = cipherText.charAt(i);

if (Character.isUpperCase(ch)) {

ch = (char) (((ch - 'A' - key + 26) % 26) + 'A');

} else {

ch = (char) (((ch - 'a' - key + 26) % 26) + 'a');

}

decryptedText+=ch;

}

System.out.println("\nDecrypted Text: " + decryptedText);

scanner.close();

}

}

**c. Hill Cipher**

**4. Write a C/JAVA program to implement the DES algorithm logic.**

import javax.crypto.\*;

public class DES{

public static void main(String[] args) {

//String we want to encrypt

String message="This is a confidential message.";

byte[] myMessage =message.getBytes(); //string to byte array as DES works on bytes

//Generating Key

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

SecretKey myDesKey = Mygenerator.generateKey();

//initializing crypto algorithm

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

//setting encryption mode

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

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

//setting decryption mode

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

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

String encrypteddata=new String(myEncryptedBytes);

String decrypteddata=new String(myDecryptedBytes);

System.out.println("Message : "+ message);

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

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

}

}

**5. Write a C/JAVA program to implement the Blowfish algorithm logic.**

//cyptro- KeyGenerator,SecretKey

import javax.crypto.\*;

//import javax.crypto.spec.SecretKeySpec;

import java.util.\*;

public class BlowFish1

{

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

{

Scanner sc=new Scanner(System.in);

KeyGenerator kgen = KeyGenerator.getInstance("Blowfish");

Cipher cipher = Cipher.getInstance("Blowfish");

SecretKey skey = kgen.generateKey();

//byte[] raw=skey.getEncoded();

//SecretKeySpec skeyspec=new SecretKeySpec(raw,"Blowfish");

//change skey to skeyspec

cipher.init(Cipher.ENCRYPT\_MODE,skey);

System.out.println("Input your message: ");

String inputText = sc.nextLine();

byte[] encrypted = cipher.doFinal(inputText.getBytes());

cipher.init(Cipher.DECRYPT\_MODE,skey);

byte[] decrypted = cipher.doFinal(encrypted);

System.out.println( "\nEncrypted text: " + new String(encrypted) + "\n" + "\nDecrypted text: "+ new String(decrypted));

}

}

**6. Write a C/JAVA program to implement the Rijndael algorithm logic.**

import javax.crypto.\*;

import javax.crypto.spec.\*;

public class AES {

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

{

String message = "Hello";

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

kgen.init(128);

SecretKey skey = kgen.generateKey();

byte[] raw = skey.getEncoded();

SecretKeySpec skeySpec = new SecretKeySpec(raw,"AES");

Cipher cipher = Cipher.getInstance("AES");

cipher.init(Cipher.ENCRYPT\_MODE,skeySpec);

byte[] encrypted = cipher.doFinal(message.getBytes());

cipher.init(Cipher.DECRYPT\_MODE, skeySpec);

byte[] decrypted = cipher.doFinal(encrypted);

String encryptedData = new String(encrypted);

String decryptedData = new String(decrypted);

System.out.println("Message:"+message);

System.out.println("Cipher Text:"+encryptedData);

System.out.println("Decrypted Text:"+decryptedData);

}

}

**7. Write the RC4 logic in Java Using Java cryptography;**

**8. Write a Java program to implement 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;

int msg = 88;

double c;

BigInteger msgback;

Scanner sc= new Scanner(System.in);

System.out.println("Enter the values of p & q: ");

p=sc.nextInt();

q=sc.nextInt();

n = p \* q;

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

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

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

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

break;

}

}

System.out.println("\nthe 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);

BigInteger N = BigInteger.valueOf(n);

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);

}

}

**9. Implement the Diffie-Hellman Key Exchange mechanism using HTML and JavaScript.**

***Diffie.html***

<html>

<head><title>Diffie Helman</title></head>

<body>

<script src="script.js"></script>

<button onclick=Exchange()>Key Exchange</button>

<div id="output"></div>

</body>

</html>

***script.js***

const p=23

const q=5

let pua,pub,pra,prb //pu-public,pr-private keys

function Exchange()

{

pra=Math.floor(Math.random()\*(p-1))+1

prb=Math.floor(Math.random()\*(p-1))+1

pua=(q\*\*pra)%p

pub=(q\*\*prb)%p

const sa=(pub\*\*pra)%p //sa-shared secret key

const sb=(pua\*\*prb)%p

document.getElementById('output').innerHTML="Private key for A: "+pra+"<br>Public key for A: "+pua+"<br>Private key for B: "+prb+"<br>Public key for B: "+pub+"<br>Shared Secret key for A: "+sa+"<br>Shared secret key for B: "+sb

}

**10. Calculate the message digest of a text using the SHA-1 algorithm in JAVA.**

import java.security.\*;

public class SHA1 {

public static void main(String args[])

{

try

{

MessageDigest md = MessageDigest.getInstance("SHA1");

String input = "Hello";

md.update(input.getBytes());

byte[] output = md.digest();

System.out.println("SHA1"+" "+input+" is\n"+bytesToHex(output));

}

catch(Exception e) {

e.printStackTrace();

}

}

public static String bytesToHex(byte[] bytes)

{

String hex = "";

for(byte b:bytes)

hex += String.format("%02X", b);

return hex;

}

}

**11. Calculate the message digest of a text using the MD-5 algorithm in JAVA.**

import java.security.\*;

public class MD5 {

public static void main(String args[])

{

try

{

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

String input = "Hello";

md.update(input.getBytes());

byte[] output = md.digest();

System.out.println("MD5"+" "+input+" is\n"+bytesToHex(output));

}

catch(Exception e) {

e.printStackTrace();

}

}

public static String bytesToHex(byte[] bytes)

{

String hex = "";

for(byte b:bytes)

hex += String.format("%02X", b);

return hex;

}

}

**12. Write a Java program to implement Poly alphabetic algorithm.**

import java.util.\*;

class PolyChiper1 {

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;

}

static String cipherText(String str, String key) {

StringBuilder cipher\_text = new StringBuilder();

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

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

x += 'A';

cipher\_text.append((char) (x));

}

return cipher\_text.toString();

}

static String originalText(String cipher\_text, String key) {

StringBuilder orig\_text = new StringBuilder();

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

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

x += 'A';

orig\_text.append((char) (x));

}

return orig\_text.toString();

}

public static void main(String[] args) {

Scanner sc=new Scanner(System.in);

String str = sc.nextLine();

String keyword = sc.nextLine();

str = str.toUpperCase();

keyword = keyword.toUpperCase();

String key = generateKey(str, keyword);

String cipher\_text = cipherText(str, key);

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

System.out.println("Original/Decrypted Text : " + originalText(cipher\_text, key));

}

}

**13. Write a Java program to implement One time Pad algorithm.**

import java.io.\*;

import java.util.\*;

public class OneTimePad {

public static String stringEncryption(String text,String key)

{

String cipherText = "";

int cipher[] = new int[key.length()];

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

cipher[i] = text.charAt(i) - 'A'+ key.charAt(i) - 'A';

}

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

if (cipher[i] > 25) {

cipher[i] = cipher[i] - 26;

}

}

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

int x = cipher[i] + 'A';

cipherText += (char)x;

}

return cipherText;

}

public static String stringDecryption(String s,String key)

{

String plainText = "";

int plain[] = new int[key.length()];

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

plain[i]= s.charAt(i) - 'A'- (key.charAt(i) - 'A');

}

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

if (plain[i] < 0) {

plain[i] = plain[i] + 26;

}

}

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

int x = plain[i] + 'A';

plainText += (char)x;

}

return plainText;

}

public static void main(String[] args)

{

Scanner sc=new Scanner(System.in);

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

String plainText = sc.nextLine();

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

String key = sc.nextLine();

String encryptedText = stringEncryption(plainText.toUpperCase(), key.toUpperCase());

System.out.println("Cipher Text - "+ encryptedText);

System.out.println("Message - "+ stringDecryption(encryptedText,key.toUpperCase()));

}

}