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.**

**PROGRAM:**

#include<stdlib.h>

main()

{

char str[]="Hello World";

char str1[11];

int i,len;

len=strlen(str);

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

{

str1[i]=str[i]^0;

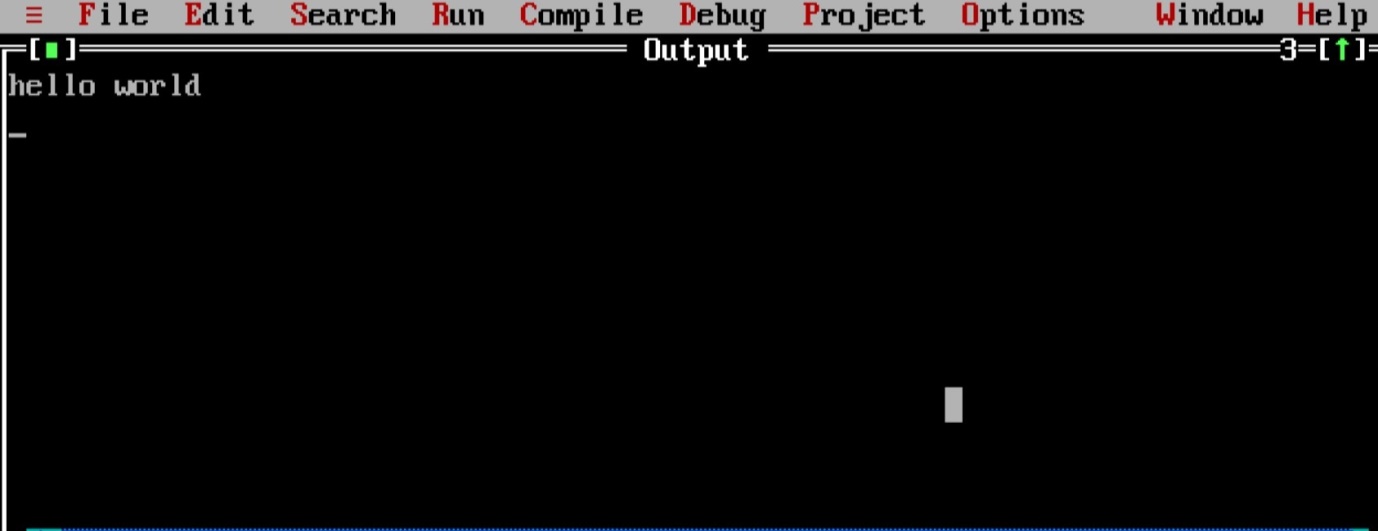
printf("%c",str1[i]);

}

printf("\n");

}

**OUTPUT:**

****

1. **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.**

**PROGRAM:**

#include <stdio.h>

#include<stdlib.h>

void main()

{

char str[]="Hello World";

char str1[11];

char str2[11]=str[];

int i,len;

len = strlen(str);

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

{

str1[i] = str[i]&127;

printf("%c",str1[i]);

}

printf("\n");

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

{

str3[i] = str2[i]^127;

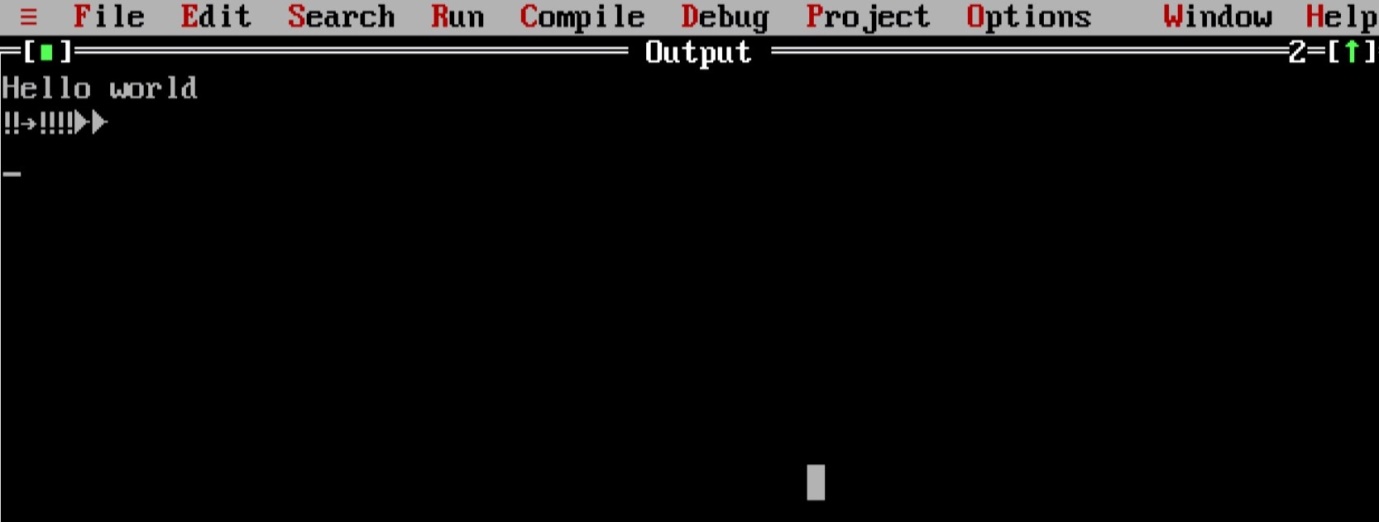
printf("%c",str3[i]);

}

printf("\n");

}

**OUTPUT:**

****

1. **Write a java program to perform encryption and decryption using the following algorithms**

**A.Caesar cipher**

**PROGRAM:**

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.util.Scanner;

public class CeaserCipher

{

static Scanner sc=new Scanner(System.in);

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

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

{

System.out.println("enter any string:");

String str=br.readLine();

System.out.println("\n Enter the key:");

int key=sc.nextInt();

String encrypted=encrypt(str,key);

System.out.println("\n ENCRYPTED STRING IS:"+encrypted);

String decrypted=decrypt(encrypted,key);

System.out.println("\n Decrypted String is:"+decrypted);

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

}

public static String encrypt(String str,int key)

{

String encrypted="";

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

{

int c=str.charAt(i);

if(Character.isUpperCase(c))

{

c=c+(key%26);

if(c>'Z')

c=c-26;

}

else if(Character.isLowerCase(c))

{

c=c+(key%26);

if(c>'z')

c=c-26;

}

encrypted+=(char)c;

}

return encrypted;

}

public static String decrypt(String str,int key)

{

String decrypted="";

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

{

int c=str.charAt(i);

if(Character.isUpperCase(c))

{

c=c-(key%26);

if(c<'A')

c=c+26;

}

else if(Character.isLowerCase(c))

{

c=c-(key%26);

if(c<'a')

c=c+26;

}

decrypted+=(char)c;

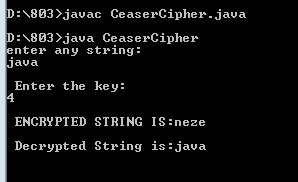
}

return decrypted;

}

}

**OUTPUT:**

****

**B.Substitution Cipher**

**PROGRAM:**

import java.io.\*;

import java.util.\*;

public class SubstitutionCipher

{

static Scanner sc=new Scanner(System.in);

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

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

{

String a="abcdefghijklmnopqrstuvwxyz";

String b="zyxwvutsrqponmlkjihgfedcba";

System.out.println("Enter any string:");

String str=br.readLine();

String decrypt="";

char c;

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

{

c=str.charAt(i);

int j=a.indexOf(c);

decrypt=decrypt+b.charAt(j);

{

System.out.println("the encrypted data is:"+decrypt);

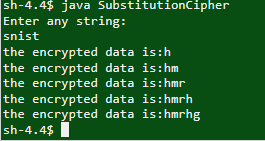
}

}

}

}

**OUTPUT:**

****

**C.Hill Cipher**

**PROGRAM:**

import java.util.\*;

import java.io.\*;

public class HillCipher

{

static float[][] decrypt = new float[3][1];

static float[][] a = new float[3][3];

static float[][] b = new float[3][3];

static float[][] mes = new float[3][1];

static float[][] res = new float[3][1];

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

static Scanner sc = new Scanner(System.in);

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

// TODO code application logic here

getkeymes();

for(int i=0;i<3;i++)

for(int j=0;j<1;j++)

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

{

res[i][j]=res[i][j]+a[i][k]\*mes[k][j];

}

System.out.print("\nEncrypted string is : ");

for(int i=0;i<3;i++)

{

System.out.print((char)(res[i][0]%26+97));

res[i][0]=res[i][0];

}

inverse();

for(int i=0;i<3;i++)

for(int j=0;j<1;j++)

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

{

decrypt[i][j] = decrypt[i][j]+b[i][k]\*res[k][j];

}

System.out.print("\nDecrypted string is : ");

for(int i=0;i<3;i++)

{

System.out.print((char)(decrypt[i][0]%26+97));

}

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

}

public static void getkeymes() throws IOException {

System.out.println("Enter 3x3 matrix for key (It should be inversible): ");

for(int i=0;i<3;i++)

for(int j=0;j<3;j++)

a[i][j] = sc.nextFloat();

System.out.print("\nEnter a 3 letter string: ");

String msg = br.readLine();

for(int i=0;i<3;i++)

mes[i][0] = msg.charAt(i)-97;

}

public static void inverse() {

float p,q;

float [][] c = a;

for(int i=0;i<3;i++)

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

//a[i][j]=sc.nextFloat();

if(i==j)

b[i][j]=1;

else b[i][j]=0;

}

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

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

p = c[i][k];

q = c[k][k];

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

if(i!=k) {

c[i][j] = c[i][j]\*q-p\*c[k][j];

b[i][j] = b[i][j]\*q-p\*b[k][j];

} } } }

for(int i=0;i<3;i++)

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

b[i][j] = b[i][j]/c[i][i]; }

System.out.println("");

System.out.println("\nInverse Matrix is : ");

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

for(int j=0;j<3;j++)

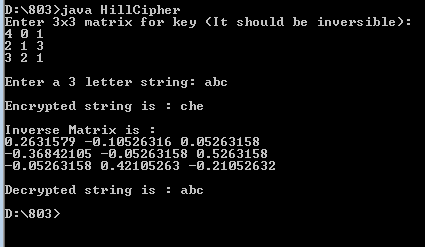
System.out.print(b[i][j] + " ");

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

}

}

**OUTPUT:**

****

1. **Write a java program to implement the Des algorithmic logic.**

**PROGRAM:**

import javax.swing.\*;

import java.security.SecureRandom;

import javax.crypto.Cipher;

import javax.crypto.KeyGenerator;

import javax.crypto.SecretKey;

import javax.crypto.spec.SecretKeySpec;

import java.util.Random ;

class DES {

byte[] skey = new byte[1000];

String skeyString;

static byte[] raw;

String inputMessage,encryptedData,decryptedMessage;

public DES() {

try {

generateSymmetricKey();

inputMessage=JOptionPane.showInputDialog(null,"Enter message to encrypt");

byte[] ibyte = inputMessage.getBytes();

byte[] ebyte=encrypt(raw, ibyte);

String encryptedData = new String(ebyte);

System.out.println("Encrypted message "+encryptedData);

JOptionPane.showMessageDialog(null,"Encrypted Data "+"\n"+encryptedData);

byte[] dbyte= decrypt(raw,ebyte);

String decryptedMessage = new String(dbyte);

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

JOptionPane.showMessageDialog(null,"Decrypted Data "+"\n"+decryptedMessage);

}

catch(Exception e) {

System.out.println(e);

}

}

void generateSymmetricKey() {

try {

Random r = new Random();

int num = r.nextInt(10000);

String knum = String.valueOf(num);

byte[] knumb = knum.getBytes();

skey=getRawKey(knumb);

skeyString = new String(skey);

System.out.println("DES Symmetric key = "+skeyString);

}

catch(Exception e) {

System.out.println(e);

}

}

private static byte[] getRawKey(byte[] seed) throws Exception {

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

SecureRandom sr = SecureRandom.getInstance("SHA1PRNG");

sr.setSeed(seed);

kgen.init(56, sr);

SecretKey skey = kgen.generateKey();

raw = skey.getEncoded();

return raw;

}

private static byte[] encrypt(byte[] raw, byte[] clear) throws Exception {

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

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

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

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

return encrypted;

}

private static byte[] decrypt(byte[] raw, byte[] encrypted) throws Exception {

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

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

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

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

return decrypted;

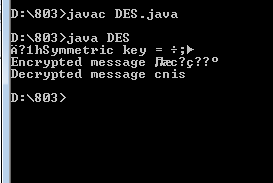
}

public static void main(String args[]) {

DES des = new DES();

}

}

**OUTPUT:**

1. **Write a java program to implement the AES algorithmic logic**

**PROGRAM:**

import java.security.\*;

import javax.crypto.\*;

import javax.crypto.spec.\*;

import java.io.\*;

import java.util.\*;

public class AES {

public static String asHex (byte buf[])

{

StringBuffer strbuf = new StringBuffer(buf.length \*2);

int i;

for (i = 0; i < buf.length; i++)

{

if (((int) buf[i] & 0xff) < 0x10)

strbuf.append("0");

strbuf.append(Long.toString((int) buf[i] & 0xff, 16));

}

return strbuf.toString();

}

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

{

Scanner sc= new Scanner(System.in);

System.out.println("enter the string");

String message=sc.nextLine();

// Get the KeyGenerator

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

kgen.init(128); // 192 and 256 bits may not be available

// Generate the secret key specs.

SecretKey skey = kgen.generateKey();

byte[] raw = skey.getEncoded();

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

// Instantiate the cipher

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

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

byte[] encrypted = cipher.doFinal((args.length == 0 ? message :args[0]).getBytes());

System.out.println("encrypted string: " +asHex(encrypted));

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

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

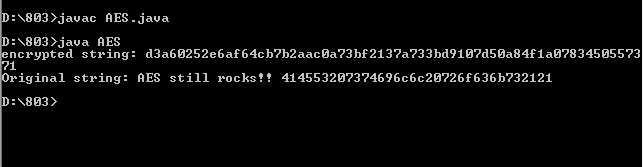
String originalString = new String(original);

System.out.println("Original string: " + originalString + " " + asHex(original));

}

}

**OUTPUT:**



**6. Write a java program to implement the Blowfish cipher algorithmic logic**

**PROGRAM:**

import javax.crypto.Cipher;

import javax.crypto.KeyGenerator;

import javax.crypto.SecretKey;

import javax.swing.JOptionPane;

public class BlowFishCipher {

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

{

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

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

SecretKey secretkey=keygenerator.generateKey();

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

String inputText = JOptionPane.showInputDialog("Input your message:");

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

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

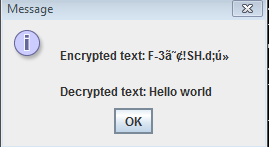
byte[] decrypted = cipher.doFinal(encrypted);

JOptionPane.showMessageDialog(JOptionPane.getRootFrame(),"\nEncrypted text: " + new String(encrypted) + "\n" + "\nDecrypted text: " + new String(decrypted));

System.exit(0);

}

}

**OUTPUT:**

**7.Write a java program to implement RSA algorithm**

**PROGRAM:**

import java.util.\*;

import java.math.\*;

class RSA1

{

public static void main(String args[])

{

Scanner sc=new Scanner(System.in);

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

System.out.println("Enter the number to be encrypted and decrypted");

int msg=sc.nextInt();

double c;

BigInteger msgback;

System.out.println("Enter 1st prime number p");

p=sc.nextInt();

System.out.println("Enter 2nd prime number q");

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) // e is for public key exponent

{

break;

}

}

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

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

{

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

if(x%e==0) //d is for private key exponent

{

d=x/e;

break;

}

}

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

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

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

System.out.println(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("Derypted message is : -");

System.out.println(msgback);

}

static int gcd(int e, int z)

{

if(e==0)

return z;

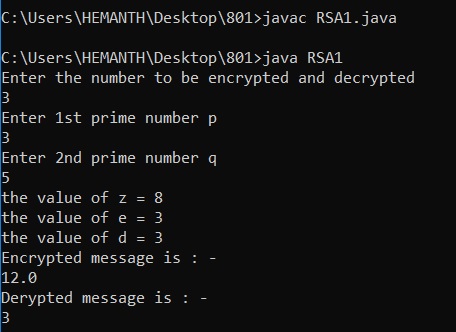
else

return gcd(z%e,e);

}

}

**OUTPUT:**

****

**8. Calculate the message digest of a text using the MD5 algorithm in JAVA.**

**PROGRAM:**

import java.security.\*;

public class MD5

{

public static void main(String[] a)

{

try

{

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

System.out.println("Message digest object info: ");

System.out.println(" Algorithm = " +md.getAlgorithm());

System.out.println(" Provider = " +md.getProvider());

System.out.println(" ToString = " +md.toString());

String input = "";

md.update(input.getBytes());

byte[] output = md.digest();

System.out.println();

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

input = "abc";

md.update(input.getBytes());

output = md.digest();

System.out.println();

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

input = "abcdefghijklmnopqrstuvwxyz";

md.update(input.getBytes());

output = md.digest();

System.out.println();

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

System.out.println("");

}

catch (Exception e) { System.out.println("Exception: " +e); }

}

public static String bytesToHex(byte[] b)

{

char hexDigit[] = {'0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'A', 'B', 'C', 'D', 'E', 'F'};

StringBuffer buf = new StringBuffer();

for (int j=0; j<b.length; j++)

{

buf.append(hexDigit[(b[j] >> 4) & 0x0f]);

buf.append(hexDigit[b[j] & 0x0f]);

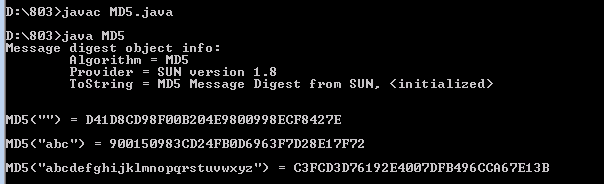
}

return buf.toString();

}

}

**OUTPUT:**

****

**9. Write the RC4 logic in Java**

**PROGRAM:**

import javax.crypto.Cipher;

import javax.crypto.spec.SecretKeySpec;

import javax.xml.bind.DatatypeConverter;

public class ArcFour

{

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

{

byte [] key = "MYVERYINSECUREKEY".getBytes("ASCII");

String clearText = "123456789012";

Cipher rc4 = Cipher.getInstance("RC4");

SecretKeySpec rc4Key = new SecretKeySpec(key, "RC4");

rc4.init(Cipher.ENCRYPT\_MODE, rc4Key);

byte [] cipherText = rc4.update(clearText.getBytes("ASCII"));

System.out.println("clear (ascii) " + clearText);

System.out.println("clear (hex) " + DatatypeConverter.printHexBinary(clearText.getBytes("ASCII")));

System.out.println("cipher (hex) is " + DatatypeConverter.printHexBinary(cipherText));

Cipher rc4Decrypt = Cipher.getInstance("RC4");

rc4Decrypt.init(Cipher.DECRYPT\_MODE, rc4Key);

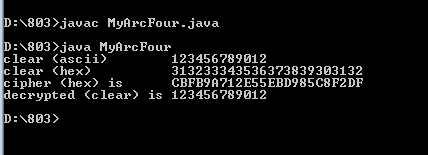
byte [] clearText2 = rc4Decrypt.update(cipherText);

System.out.println("decrypted (clear) is " + new String(clearText2, "ASCII"));

}

}

**OUTPUT:**

****

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

**PROGRAM:**

import java.security.\*;

public class SHA1

{

public static void main(String[] a)

{

try {

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

System.out.println("Message digest object info: ");

System.out.println(" Algorithm = " +md.getAlgorithm());

System.out.println(" Provider = " +md.getProvider());

System.out.println(" ToString = " +md.toString());

String input = "";

md.update(input.getBytes());

byte[] output = md.digest();

System.out.println();

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

input = "abc";

md.update(input.getBytes());

output = md.digest();

System.out.println();

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

input = "abcdefghijklmnopqrstuvwxyz";

md.update(input.getBytes());

output = md.digest();

System.out.println();

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

System.out.println("");

}

catch (Exception e)

{

System.out.println("Exception: " +e);

}

}

public static String bytesToHex(byte[] b)

{

char hexDigit[] = {'0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'A', 'B', 'C', 'D', 'E', 'F'};

StringBuffer buf = new StringBuffer();

for (int j=0; j<b.length; j++)

{

buf.append(hexDigit[(b[j] >> 4) & 0x0f]);

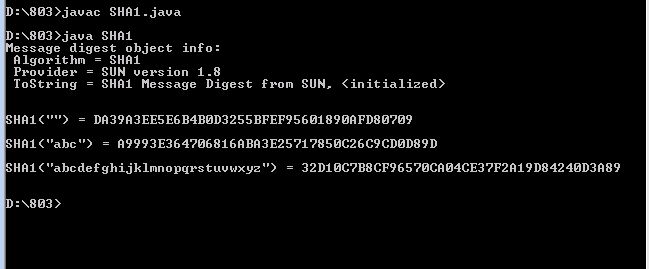
buf.append(hexDigit[b[j] & 0x0f]);

}

return buf.toString();

}

}

**OUTPUT:**

**11. Implement the Diffie-Hellman Key Exchange mechanism using JAVA**

**PROGRAM:**

import java.math.BigInteger;

import java.security.KeyFactory;

import java.security.KeyPair;

import java.security.KeyPairGenerator;

import java.security.SecureRandom;

import javax.crypto.spec.DHParameterSpec;

import javax.crypto.spec.DHPublicKeySpec;

public class DiffeHellman

{

public final static int pValue = 47;

public final static int gValue = 71;

public final static int XaValue = 9;

public final static int XbValue = 14;

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

{

BigInteger p = new BigInteger(Integer.toString(pValue));

BigInteger g = new BigInteger(Integer.toString(gValue));

BigInteger Xa = new BigInteger(Integer.toString(XaValue));

BigInteger Xb = new BigInteger(Integer.toString(XbValue));

createKey();

int bitLength = 512;

SecureRandom rnd = new SecureRandom();

p = BigInteger.probablePrime(bitLength, rnd);

g = BigInteger.probablePrime(bitLength, rnd);

createSpecificKey(p, g);

}

public static void createKey() throws Exception

{

KeyPairGenerator kpg = KeyPairGenerator.getInstance("DiffieHellman");

kpg.initialize(512);

KeyPair kp = kpg.generateKeyPair();

KeyFactory kfactory = KeyFactory.getInstance("DiffieHellman");

DHPublicKeySpec kspec = (DHPublicKeySpec) kfactory.getKeySpec(kp.getPublic(), DHPublicKeySpec.class);

System.out.println("Public key is: " +kspec);

}

public static void createSpecificKey(BigInteger p, BigInteger g) throws Exception

{

KeyPairGenerator kpg = KeyPairGenerator.getInstance("DiffieHellman");

DHParameterSpec param = new DHParameterSpec(p, g);

kpg.initialize(param);

KeyPair kp = kpg.generateKeyPair();

KeyFactory kfactory = KeyFactory.getInstance("DiffieHellman");

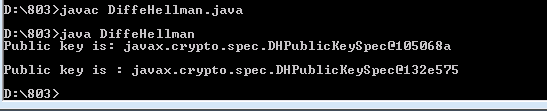
DHPublicKeySpec kspec = (DHPublicKeySpec) kfactory.getKeySpec(kp.getPublic(), DHPublicKeySpec.class);

System.out.println("\nPublic key is : " +kspec);

}

}

**OUTPUT:**

****

12. **DSA Key generation (public and private key pair) can be performed using java. Write a program which can do this.**

**PROGRAM:**

import java.security.KeyPair;

import java.security.KeyPairGenerator;

import java.security.PrivateKey;

import java.security.PublicKey;

public class DSAKeyGen

{

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

{

KeyPairGenerator keyGen = KeyPairGenerator.getInstance("DSA");

keyGen.initialize(1024);

KeyPair keypair = keyGen.genKeyPair();

PrivateKey privateKey = keypair.getPrivate();

System.out.println(privateKey);

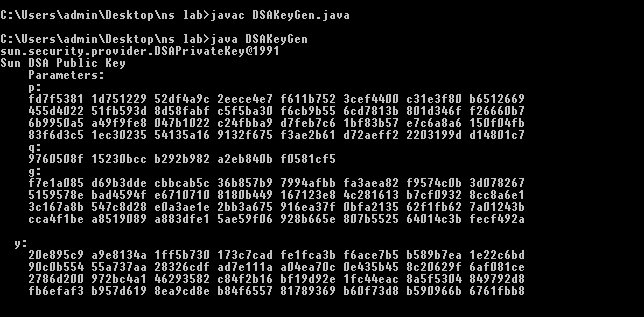
PublicKey publicKey = keypair.getPublic();

System.out.println(publicKey);

}

}

**OUTPUT:**



**13**. **Write a program in java, which performs a Digital signature on a given text.**

**PROGRAM:**

import java.security.KeyPair;

import java.security.KeyPairGenerator;

import java.security.Signature;

import sun.misc.BASE64Encoder;

public class DigSign {

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

// TODO code application logic here

KeyPairGeneratorkpg = KeyPairGenerator.getInstance("RSA");

kpg.initialize(1024);

KeyPairkeyPair = kpg.genKeyPair();

byte[] data = "Sample Text".getBytes("UTF8");

Signature sig = Signature.getInstance("MD5WithRSA");

sig.initSign(keyPair.getPrivate());

sig.update(data);

byte[] signatureBytes = sig.sign();

System.out.println("Signature: \n" + new BASE64Encoder().encode(signatureBytes));

sig.initVerify(keyPair.getPublic());

sig.update(data);

System.out.println(sig.verify(signatureBytes));

}

}

**OUTPUT:**