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

**a) Ceaser Cipher**

**b) Substitution Cipher**

**c) Hill Cipher**

**a)CeaserCipher**

**CODE:**

**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.print("Enter any String: ");**

**String str = br.readLine();**

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

**int key = sc.nextInt();**

**String encrypted = encrypt(str, key);**

**System.out.println("\nEncrypted String is: " +encrypted);**

**String decrypted = decrypt(encrypted, key);**

**System.out.println("\nDecrypted 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:**

**Enter any String: Hello World**

**Enter the Key: 5**

**Encrypted String is: MjqqtBtwqi**

**Decrypted String is: Hello World**

**b) Substitution Cipher**

**CODE:**

**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 {**

**// TODO code application logic here**

**String a = "abcdefghijklmnopqrstuvwxyz";**

**String b = "zyxwvutsrqponmlkjihgfedcba";**

**System.out.print("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:**

**Enter any string: aceho**

**The encrypted data is: zxvsl**

**C)Hill Cipher**

**CODE:**

**class hillCipher {**

**/\* 3x3 key matrix for 3 characters at once \*/**

**public static int[][] keymat = new int[][] { { 1, 2, 1 }, { 2, 3, 2 },**

**{ 2, 2, 1 } }; /\* key inverse matrix \*/**

**public static int[][] invkeymat = new int[][] { { -1, 0, 1 }, { 2, -1, 0 }, { -2, 2, -1**

**} };**

**public static String key = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";**

**private static String encode(char a, char b, char c) {**

**String ret = "";**

**int x, y, z;**

**int posa = (int) a - 65;**

**int posb = (int) b - 65;**

**int posc = (int) c - 65;**

**x = posa \* keymat[0][0] + posb \* keymat[1][0] + posc \* keymat[2][0];**

**y = posa \* keymat[0][1] + posb \* keymat[1][1] + posc \* keymat[2][1];**

**z = posa \* keymat[0][2] + posb \* keymat[1][2] + posc \* keymat[2][2];**

**a = key.charAt(x % 26);**

**b = key.charAt(y % 26);**

**c = key.charAt(z % 26);**

**ret = "" + a + b + c;**

**return ret;**

**}**

**private static String decode(char a, char b, char c) {**

**String ret = "";**

**int x, y, z;**

**int posa = (int) a - 65;**

**int posb = (int) b - 65;**

**int posc = (int) c - 65;**

**x = posa \* invkeymat[0][0] + posb \* invkeymat[1][0] + posc \***

**invkeymat[2][0];**

**y = posa \* invkeymat[0][1] + posb \* invkeymat[1][1] + posc \***

**invkeymat[2][1];**

**z = posa \* invkeymat[0][2] + posb \* invkeymat[1][2] + posc \***

**invkeymat[2][2];**

**a = key.charAt((x % 26 < 0) ? (26 + x % 26) : (x % 26));**

**b = key.charAt((y % 26 < 0) ? (26 + y % 26) : (y % 26));**

**c = key.charAt((z % 26 < 0) ? (26 + z % 26) : (z % 26));**

**ret = "" + a + b + c;**

**return ret;**

**}**

**public static void main(String[] args) throws java.lang.Exception {**

**String msg;**

**String enc = "";**

**String dec = "";**

**int n;**

**msg = ("SecurityLaboratory");**

**System.out.println("simulation of Hill Cipher\n-------------------------");**

**System.out.println("Input message : " + msg);**

**msg = msg.toUpperCase();**

**msg = msg.replaceAll("\\s", "");**

**/\* remove spaces \*/ n = msg.length() % 3;**

**/\* append padding text X \*/ if (n != 0) {**

**for (int i = 1; i <= (3 - n); i++) {**

**msg += 'X';**

**}**

**}**

**System.out.println("padded message : " + msg);**

**char[] pdchars = msg.toCharArray();**

**for (int i = 0; i < msg.length(); i += 3) {**

**enc += encode(pdchars[i], pdchars[i + 1], pdchars[i + 2]);**

**}**

**System.out.println("encoded message : " + enc);**

**char[] dechars = enc.toCharArray();**

**for (int i = 0; i < enc.length(); i += 3) {**

**dec += decode(dechars[i], dechars[i + 1], dechars[i + 2]);**

**}**

**System.out.println("decoded message : " + dec);**

**}**

**}**

**OUTPUT:**

**simulation of Hill Cipher**

**-------------------------**

**Input message : SecurityLaboratory**

**padded message : SECURITYLABORATORY**

**encoded message : EACSDKLCAEFQDUKSXU**

**decoded message : SECURITYLABORATORY**

**2.Write a Java program to implement the DES algorithm logic.**

**CODE:**

**import java.security.InvalidKeyException;**

**import java.security.NoSuchAlgorithmException;**

**import javax.crypto.BadPaddingException;**

**import javax.crypto.Cipher;**

**import javax.crypto.IllegalBlockSizeException;**

**import javax.crypto.KeyGenerator;**

**import javax.crypto.NoSuchPaddingException;**

**import javax.crypto.SecretKey;**

**public class DES**

**{**

**public static void main(String[] argv)**

**{**

**try{**

**System.out.println("Message Encryption Using DES Algorithm\n-------");**

**KeyGenerator keygenerator = KeyGenerator.getInstance("DES");**

**SecretKey myDesKey = keygenerator.generateKey();**

**Cipher desCipher;**

**desCipher = Cipher.getInstance("DES/ECB/PKCS5Padding");**

**desCipher.init(Cipher.ENCRYPT\_MODE, myDesKey);**

**byte[] text = "Secret Information ".getBytes();**

**System.out.println("Message [Byte Format] : " + text);**

**System.out.println("Message : " + new String(text));**

**byte[] textEncrypted = desCipher.doFinal(text);**

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

**desCipher.init(Cipher.DECRYPT\_MODE, myDesKey);**

**byte[] textDecrypted = desCipher.doFinal(textEncrypted);**

**System.out.println("Decrypted Message: " + new**

**String(textDecrypted));**

**}catch(NoSuchAlgorithmException e){**

**e.printStackTrace();**

**}catch(NoSuchPaddingException e){**

**e.printStackTrace();**

**}catch(InvalidKeyException e){**

**e.printStackTrace();**

**}catch(IllegalBlockSizeException e){**

**e.printStackTrace();**

**}catch(BadPaddingException e){**

**e.printStackTrace();**

**}**

**}**

**}**

**OUTPUT:**

**3.Implementation of symmetric cipher algorithm(AES) in java.**

**CODE:**

**import java.io.UnsupportedEncodingException;**

**import java.security.MessageDigest;**

**import java.security.NoSuchAlgorithmException;**

**import java.util.Arrays;**

**import java.util.Base64;**

**import javax.crypto.Cipher;**

**import javax.crypto.spec.SecretKeySpec;**

**public class AES {**

**private static SecretKeySpec secretKey;**

**private static byte[] key;**

**public static void setKey(String myKey) {**

**MessageDigest sha = null;**

**try {**

**key = myKey.getBytes("UTF-8");**

**sha = MessageDigest.getInstance("SHA-1");**

**key = sha.digest(key);**

**key = Arrays.copyOf(key, 16);**

**secretKey = new SecretKeySpec(key, "AES");**

**} catch (NoSuchAlgorithmException e) {**

**e.printStackTrace();**

**} catch (UnsupportedEncodingException e) {**

**e.printStackTrace();**

**}**

**}**

**public static String encrypt(String strToEncrypt, String secret) {**

**try {**

**setKey(secret);**

1. **Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");**

**cipher.init(Cipher.ENCRYPT\_MODE, secretKey);**

**return**

**Base64.getEncoder().encodeToString(cipher.doFinal(strToEncrypt.getBytes("UTF-8")));**

**} catch (Exception e) {**

**System.out.println("Error while encrypting: " + e.toString());**

**}**

**return null;**

**}**

**public static String decrypt(String strToDecrypt, String secret) {**

**try {**

**setKey(secret);**

**Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5PADDING");**

**cipher.init(Cipher.DECRYPT\_MODE, secretKey);**

**return new**

**String(cipher.doFinal(Base64.getDecoder().decode(strToDecrypt)));**

**} catch (Exception e) {**

**System.out.println("Error while decrypting: " + e.toString());**

**}**

**return null;**

**}**

**public static void main(String[] args) {**

**final String secretKey = "annaUniversity";**

**String originalString = "www.annauniv.edu";**

**String encryptedString = AES.encrypt(originalString, secretKey);**

**String decryptedString = AES.decrypt(encryptedString, secretKey);**

**System.out.println("URL Encryption Using AES Algorithm\n------------");**

**System.out.println("Original URL : " + originalString);**

**System.out.println("Encrypted URL : " + encryptedString);**

**System.out.println("Decrypted URL : " + decryptedString);**

**}**

**}**

**OUTPUT:**

**URL Encryption Using AES Algorithm**

**-------------------------------------------------**

**Original URL :** [**www.annauniv.edu**](http://www.annauniv.edu)

**Encrypted URL : vibpFJW6Cvs5Y+L7t4N6YWWe07+JzS1d3CU2h3mEvEg=**

**Decrypted URL : www.annauniv.edu**

**4. write a java program to implement the BlowFish algorithm logic.**

**CODE:**

**import java.io.UnsupportedEncodingException;**

**import java.nio.charset.Charset;**

**import java.security.InvalidKeyException;**

**import java.security.NoSuchAlgorithmException;**

**import java.util.Base64;**

**import javax.crypto.BadPaddingException;**

**import javax.crypto.Cipher;**

**import javax.crypto.IllegalBlockSizeException;**

**import javax.crypto.NoSuchPaddingException;**

**import javax.crypto.spec.SecretKeySpec;**

**/\*\***

**\* This program demonstrates how to encrypt/decrypt**

**\* input using the Blowfish**

**\* Cipher with the Java Cryptography.**

**\*/**

**public class BlowfishDemo {**

**public String encrypt(String password, String key) throws**

**NoSuchAlgorithmException, NoSuchPaddingException,**

**InvalidKeyException, IllegalBlockSizeException,**

**BadPaddingException, UnsupportedEncodingException {**

**byte[] KeyData = key.getBytes();**

**SecretKeySpec KS = new SecretKeySpec(KeyData, "Blowfish");**

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

**cipher.init(Cipher.ENCRYPT\_MODE, KS);**

**Stri ng encryptedtext = Base64.getEncoder().**

**encodeToString(cipher.doFinal(password.getBytes("UTF-8")));**

**return encryptedtext;**

**}**

**public String decrypt(String encryptedtext, String key)**

**throws NoSuchAlgorithmException, NoSuchPaddingException,**

**InvalidKeyException, IllegalBlockSizeException,**

**BadPaddingException {**

**byte[] KeyData = key.getBytes();**

**SecretKeySpec KS = new SecretKeySpec(KeyData, "Blowfish");**

**byte[] ecryptedtexttobytes = Base64.getDecoder().**

**decode(encryptedtext);**

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

**cipher.init(Cipher.DECRYPT\_MODE, KS);**

**byte[] decrypted = cipher.doFinal(ecryptedtexttobytes);**

**String decryptedString =**

**new String(decrypted, Charset.forName("UTF-8"));**

**return decryptedString;**

**}**

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

**final String password = "Knf@123";**

**final String key = "knowledgefactory";**

**System.out.println("Password: " + password);**

**BlowfishDemo obj = new BlowfishDemo();**

**Strin g enc\_output = obj.encrypt(password, key);**

**System.out.println("Encrypted text: " + enc\_output);**

**String dec\_output = obj.decrypt(enc\_output, key);**

**System.out.println("Decrypted text: " + dec\_output);**

**}**

**}**

**OUTPUT:**

**Password: Knf@123**

**Encrypted text: 4DTHqnctCuk=**

**Decrypted text: Knf@123**

**5.Write a Java Program to implement RSA algorithm.**

**CODE:**

**import java.util.\*;**

**import java.math.\*;**

**class RSA**

**{**

**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;**

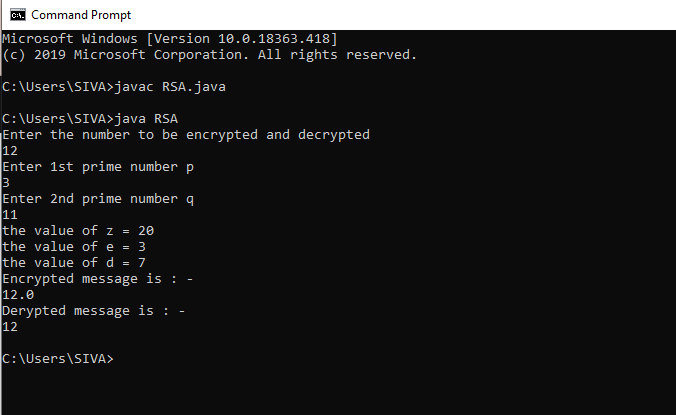
**else**

**return gcd(z%e,e);**

**}**

**}**

**OUTPUT:**

****

**6.Implement the Diffie-Hellman Key Exchange mechanism using html and java script. Consider the end user as one of the parties(Alice) and the java script application as the other party(Bob).**

**CODE:**

**import java.util.\*;**

**class Diffie\_Hellman**

**{**

**public static void main(String args[])**

**{**

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

**System.out.println("Enter modulo(p)");**

**int p=sc.nextInt();**

**System.out.println("Enter primitive root of "+p);**

**int g=sc.nextInt();**

**System.out.println("Choose 1st secret no(Alice)");**

**int a=sc.nextInt();**

**System.out.println("Choose 2nd secret no(BOB)");**

**int b=sc.nextInt();**

**int A = (int)Math.pow(g,a)%p;**

**int B = (int)Math.pow(g,b)%p;**

**int S\_A = (int)Math.pow(B,a)%p;**

**int S\_B =(int)Math.pow(A,b)%p;**

**if(S\_A==S\_B)**

**{**

**System.out.println("ALice and Bob can communicate with each other!!!");**

**System.out.println("They share a secret no = "+S\_A);**

**}**

**else**

**{**

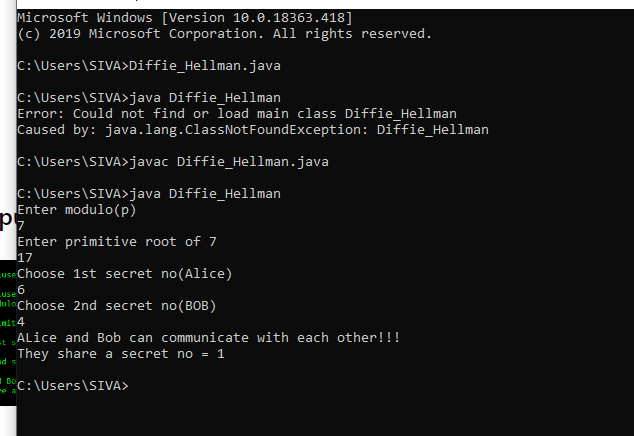
**System.out.println("ALice and Bob cannot communicate with each other!!!");**

**}**

**}**

**}**

**OUTPUT:**

****

**8.Implement ELGAMAL cryptosystem.**

**CODE:**

**import java.math.\*;**

**import java.util.\*;**

**import java.security.\*;**

**import java.io.\*;**

**public class ElGamal**

**{**

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

**{**

**BigInteger p, b, c, secretKey;**

**Random sc = new SecureRandom();**

**secretKey = new BigInteger("12345678901234567890");**

**//**

**// public key calculation**

**//**

**System.out.println("secretKey = " + secretKey);**

**p = BigInteger.probablePrime(64, sc);**

**b = new BigInteger("3");**

**c = b.modPow(secretKey, p);**

**System.out.println("p = " + p);**

**System.out.println("b = " + b);**

**System.out.println("c = " + c);**

**//**

**// Encryption**

**//**

**System.out.print("Enter your Big Number message -->");**

**String s = Tools.getString();**

**BigInteger X = new BigInteger(s);**

**BigInteger r = new BigInteger(64, sc);**

**BigInteger EC = X.multiply(c.modPow(r, p)).mod(p);**

**BigInteger brmodp = b.modPow(r, p);**

**System.out.println("Plaintext = " + X);**

**System.out.println("r = " + r);**

**System.out.println("EC = " + EC);**

**System.out.println("b^r mod p = " + brmodp);**

**//**

**// Decryption**

**//**

**BigInteger crmodp = brmodp.modPow(secretKey, p);**

**BigInteger d = crmodp.modInverse(p);**

**BigInteger ad = d.multiply(EC).mod(p);**

**System.out.println("\n\nc^r mod p = " + crmodp);**

**System.out.println("d = " + d);**

**System.out.println("Alice decodes: " + ad);**

**}**

**}**

**OUTPUT:**

**9. Authenticating the given signature using MD5 hash algoritham.**

**CODE:**

**import java.math.BigInteger;**

**import java.security.MessageDigest;**

**import java.security.NoSuchAlgorithmException;**

**public class MD5Example**

**{**

**//hash function to get the md5 hash**

**public static String getMd5Hash(String input)**

**{**

**try**

**{**

**//static getInstance() method is called with hashing MD5**

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

**//calculating message digest of an input that return array of byte**

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

**//converting byte array into signum representation**

**BigInteger no = new BigInteger(1, messageDigest);**

**//converting 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**

**{**

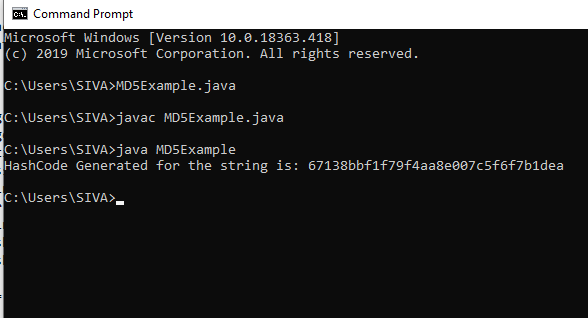
**String s = "javatpoint";**

**System.out.println("HashCode Generated for the string is: " + getMd5Hash(s));**

**}**

**}**

**OUTPUT:**

****

**9. Calculate the message digest of a text using the SHA-1 algorithm in java.**

**CODE:**

**// Java program to calculate SHA-1 hash value**

**import java.math.BigInteger;**

**import java.security.MessageDigest;**

**import java.security.NoSuchAlgorithmException;**

**public class SHA {**

**public static String encryptThisString(String input)**

**{**

**try {**

**// getInstance() method is called with algorithm SHA-1**

**MessageDigest md = MessageDigest.getInstance("SHA-1");**

**// digest() method is called**

**// to calculate message digest of the input string**

**// returned as 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);**

**// Add preceding 0s to make it 32 bit**

**while (hashtext.length() < 32) {**

**hashtext = "0" + hashtext;**

**}**

**// return the 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("HashCode Generated by SHA-1 for: ");**

**String s1 = "GeeksForGeeks";**

**System.out.println("\n" + s1 + " : " + encryptThisString(s1));**

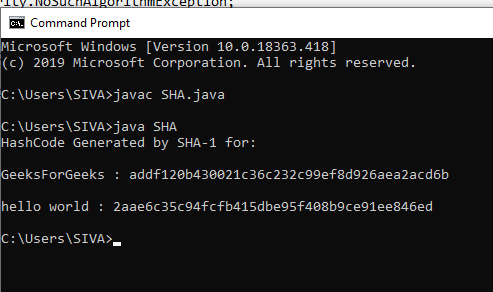
**String s2 = "hello world";**

**System.out.println("\n" + s2 + " : " + encryptThisString(s2));**

**}**

**}**

**OUTPUT:**

****