**Exercise 6 Rivest-Shamir-Adleman (RSA) Algorithm**

**CODE:**

import java.io.IOException;

import java.math.BigInteger;

import java.util.\*;

public class RSA {

private BigInteger p;

private BigInteger q;

private BigInteger N;

private BigInteger phi;

private BigInteger e;

private BigInteger d;

private int bitlength = 1024;

private Random r;

public RSA() {

r = new Random();

p = BigInteger.probablePrime(bitlength, r);

q = BigInteger.probablePrime(bitlength, r);

N = p.multiply(q);

phi = p.subtract(BigInteger.ONE).multiply(q.subtract(BigInteger.ONE));

e = BigInteger.probablePrime(bitlength / 2, r);

while (phi.gcd(e).compareTo(BigInteger.ONE) > 0 && e.compareTo(phi) < 0) //gcd(e,phi)=1 and 1<e<phi

{

e.add(BigInteger.ONE);

}

d = e.modInverse(phi);

}

public byte[] encrypt(byte[] message) {

BigInteger res = new BigInteger(message).modPow(e, N);

System.out.println("\nThe cipher text is (in Big Integer) " + res);

return res.toByteArray();

}

public byte[] decrypt(byte[] message) {

BigInteger res = new BigInteger(message).modPow(d, N);

System.out.println("\nThe plain text is (in Big Integer) " + res);

return res.toByteArray();

}

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

Scanner sc = new Scanner(System.in);

System.out.println("\nRSA ALGORITHM");

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

RSA rsa = new RSA();

String plain, cipher;

System.out.println("\nKey Generation");

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

System.out.println("\nP is (in Big Integer)");

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

System.out.println("\nQ is (in Big Integer)");

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

System.out.println("\nN is (in Big Integer)");

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

System.out.println("\nPHI (N) is (in Big Integer)");

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

System.out.println("\ne is (in Big Integer)");

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

System.out.println("\nThe private key 'd' is (in Big Integer)");

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

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

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

System.out.print("\nEnter the plain text: ");

plain = sc.nextLine();

byte[] plainB =plain.getBytes();

System.out.println(

"\nThe plain text is (in Big Integer) " + new BigInteger(plainB)

);

System.out.println(

"\nThe plain text is (in Base64) " +

Base64.getEncoder().encodeToString(plainB)

);

byte[] cipherB = rsa.encrypt(plainB);

System.out.println(

"\nThe cipher text is (in Base 64): " +

Base64.getEncoder().encodeToString(cipherB)

);

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

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

System.out.print("\nEnter the cipher text (in Base64): ");

cipher = sc.nextLine();

cipherB = Base64.getDecoder().decode(cipher);

System.out.println(

"\nThe cipher text is (in Big Integer) " + new BigInteger(cipherB)

);

plainB = rsa.decrypt(cipherB);

System.out.println(

"\nThe plain Text is (in Base 64): " +

Base64.getEncoder().encodeToString(plainB)

);

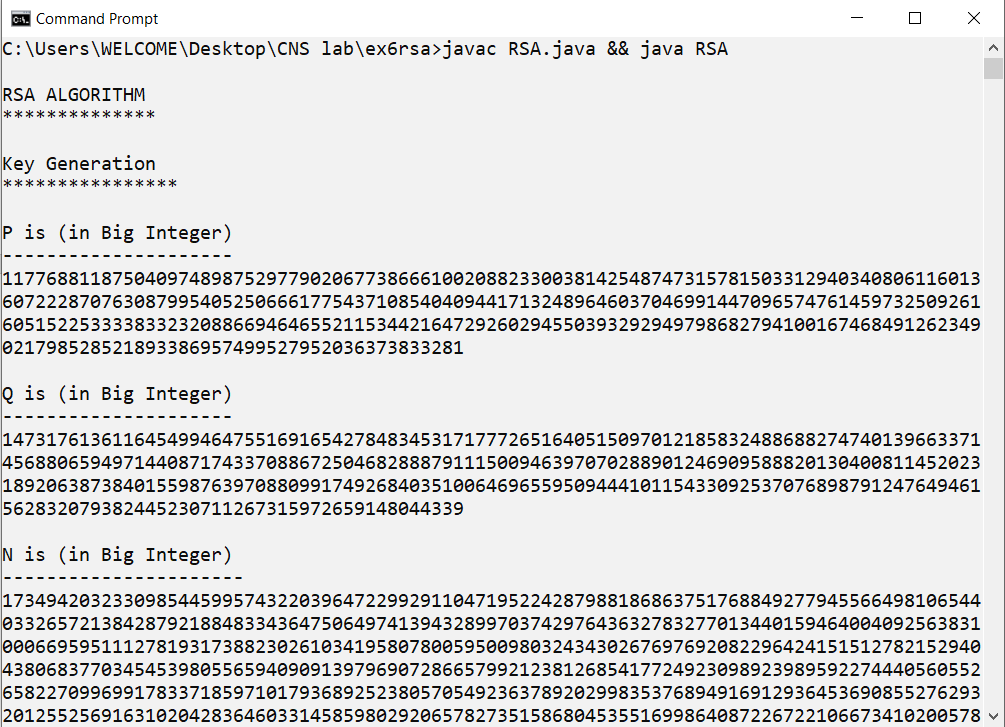
System.out.println("\nThe original plain Text is: " + new String(plainB));

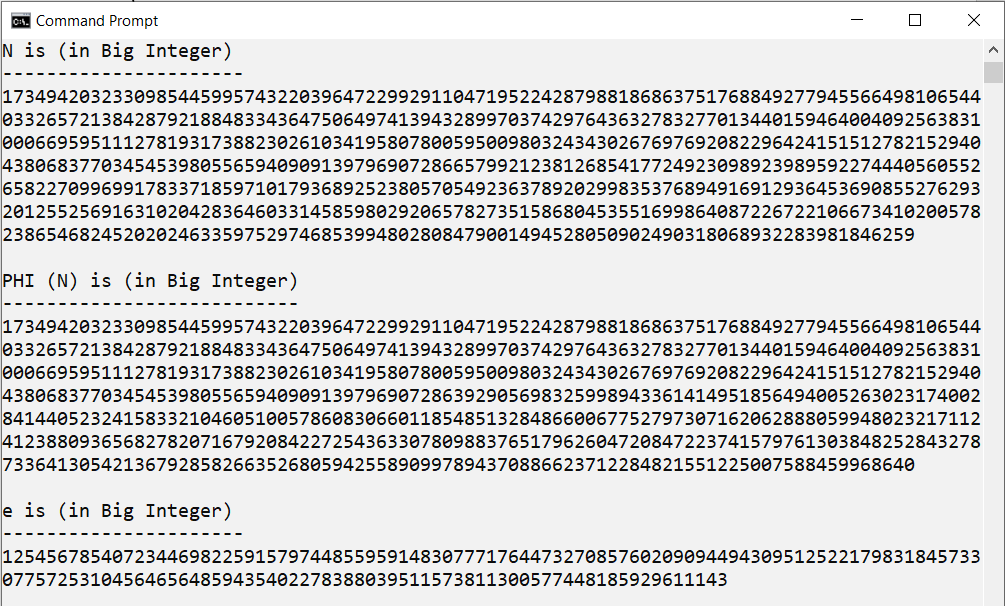
}

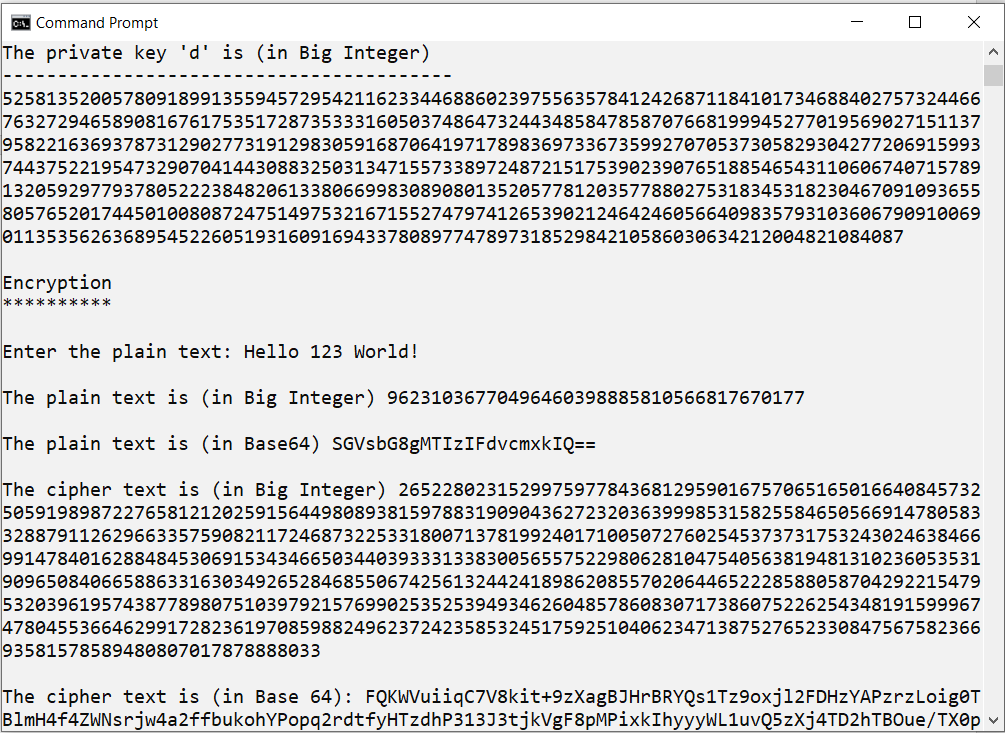
}

**OUTPUT:**

**Example 1:**









**Example 2:**

