Assignment no 4

Problem statement:
Write a Java/C/C++/Python program to implement RSA algorithm.

Roll no: 331 Name: Rameshwar Gosavi Batch: B

Solution:

```
package RSA;

import java.math.BigInteger;
import java.util.Random;
import java.util.Scanner;

public class RSAAlorithm {

	public static void main(String[] args) {

		Scanner sc = new Scanner(System.in);
		System.out.println("Enter first prime number: ");
		int p = sc.nextInt();
		System.out.println("Enter second prime number: ");
		int q = sc.nextInt();
		System.out.println("Enter message: ");
		int m = sc.nextInt();
		int primefirst = p;
		int primesecond = q;
		int message = m;
		int n = primefirst * primesecond;

		int value_Of_Fi_Of_N = fiofn(primefirst, primesecond);

		Random rand = new Random();
		int e;
		do {
			e = rand.nextInt(20) + 1;
		} while (e == 2 || e == p || e == q || !isPrime(e) || !isGCDOne(e,
value_Of_Fi_Of_N));

		System.out.println("**********************");
		displaypublickey(e, n);

		int d = displayprivatekey(e, value_Of_Fi_Of_N);
		System.out.println("D: " + d);

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

		System.out.println("Encrypted message: " + encryption(message, e, n));
		;
```

```
		decryption(d, n, encryption(message, e, n));

	}

	static public int fiofn(int p, int q) {
		int firstnum = p - 1;
		int secondnum = q - 1;
		int n = firstnum * secondnum;
		return n;
	}

	static public boolean isPrime(int n) {

		if (n <= 1) {
			return false;
		}
		for (int i = 2; i <= Math.sqrt(n); i++) {
			if (n % i == 0) {
				return false;
			}
		}
		return true;

	}

	public static boolean isGCDOne(int num1, int num2) {
		int gcd = findGCD(num1, num2);
		return gcd == 1;
	}

	public static int findGCD(int num1, int num2) {
		if (num2 == 0) {
			return num1;
		}
		return findGCD(num2, num1 % num2);
	}

	public static void displaypublickey(int e, int n) {
		System.out.println("public key: " + e + " " + n);
	}

	public static int displayprivatekey(int e, int value_Of_Fi_Of_N) {
		int d;
		for (d = 0; d < value_Of_Fi_Of_N; d++)

		{
			if ((e * d) % value_Of_Fi_Of_N == 1) {

				return d;
			}
```

```
            }
            return 0;

        }

        public static int encryption(int m, int e, int n) {
            BigInteger base = BigInteger.valueOf(m);
            BigInteger exponent = BigInteger.valueOf(e);
            BigInteger modulus = BigInteger.valueOf(n);
            BigInteger result = base.modPow(exponent, modulus);
            int e1 = result.intValue();
            return e1;

        }

        public static BigInteger decryption(int d, int n, long encryption) {
            BigInteger base = BigInteger.valueOf(encryption);
            BigInteger exponent = BigInteger.valueOf(d);
            BigInteger modulus = BigInteger.valueOf(n);
            BigInteger result = base.modPow(exponent, modulus);
            System.out.println("Decrypted message: " + result);

            return result;
        }

}
```

Output: