EXP -06

AIM: Implementation of RSA algorithm.
Code:

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

public class RSA {

    // Function to find GCD
    public static BigInteger gcd(BigInteger a, BigInteger b) {
        while (!b.equals(BigInteger.ZERO)) {
            BigInteger temp = b;
            b = a.mod(b);
            a = temp;
        }
        return a;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Random rand = new Random();

        // Input two prime numbers
        System.out.print("Enter a prime number p: ");
        BigInteger p = sc.nextBigInteger();
        System.out.print("Enter a prime number q: ");
        BigInteger q = sc.nextBigInteger();

        // Prime check
        if (!p.isProbablePrime(10)) {
            System.out.println("Error: p is not a prime number. Exiting.");
            return;
        }
        if (!q.isProbablePrime(10)) {
            System.out.println("Error: q is not a prime number. Exiting.");
            return;
        }

        // n = p * q
        BigInteger n = p.multiply(q);

        // phi = (p - 1) * (q - 1)
        BigInteger phi =
(p.subtract(BigInteger.ONE)).multiply(q.subtract(BigInteger.ONE));

        // Choose e such that 1 < e < phi and gcd(e, phi) = 1
        BigInteger e, d;
        while (true) {
            e = new BigInteger(phi.bitLength(), rand);
            if (e.compareTo(BigInteger.ONE) > 0 && e.compareTo(phi) < 0
&& gcd(e, phi).equals(BigInteger.ONE)) {
                d = e.modInverse(phi);
                if (!e.equals(d)) break; // Ensure e != d
            }
        }

        System.out.println("\nPublic Key: (" + e + ", " + n + ")");
        System.out.println("Private Key: (" + d + ", " + n + ")");

        // Consume leftover newline
        sc.nextLine();
        // Input message
        System.out.print("\nEnter the message to encrypt: ");
        String message = sc.nextLine();
        char[] chars = message.toCharArray();

        // Encrypt each character
        BigInteger[] encrypted = new BigInteger[chars.length];
        System.out.print("Encrypted Message: ");
        for (int i = 0; i < chars.length; i++) {
            BigInteger m = BigInteger.valueOf((int) chars[i]);
            encrypted[i] = m.modPow(e, n);
            System.out.print(encrypted[i] + " ");
        }

        // Decrypt
        System.out.print("\nDecrypted Message: ");
        for (BigInteger c : encrypted) {
            BigInteger m = c.modPow(d, n);
            char decryptedChar = (char) m.intValue();
            System.out.print(decryptedChar);
        }
    }
}
```