EX NO: **RSA ALGORITHM** REG NO:210701290

DATE:

AIM:-

To implement a RSA algorithm using Java.

ALGORITHM:-

STEP 1:Choose two distinct prime numbers, p and q.

STEP 2:Calculate their product, n = p * q, which will be the modulus for the public and private keys.

STEP 3:Compute the totient function of n, φ(n) = (p-1) * (q-1).

STEP 4:Choose an integer e such that 1 < e < φ(n) and gcd(e, φ(n)) = 1, which will be the public exponent.

STEP 5:Compute the modular multiplicative inverse of e modulo φ(n), which will be the private exponent d.

STEP 6:The public key is (e, n) and the private key is (d, n).

STEP 7:To encrypt a message m, compute c ≡ m^e (mod n), and to decrypt a ciphertext c, compute m ≡ c^d (mod n).

PROGRAM:-

```
import java.math.BigInteger;

import java.util.Random;

public class Main {

   public static void main(String[] args) {

      BigInteger p = BigInteger.probablePrime(6, new Random());

      BigInteger q = BigInteger.probablePrime(6, new Random());
```

```
        BigInteger n = p.multiply(q);
        BigInteger phi =
(p.subtract(BigInteger.ONE)).multiply(q.subtract(BigInteger.ONE));
        BigInteger e = BigInteger.valueOf(3); // Example public key
        while (phi.gcd(e).intValue() > 1) {
           e = e.add(BigInteger.ONE);}
        BigInteger d = e.modInverse(phi);
        BigInteger message = BigInteger.valueOf(42);
        BigInteger encrypted = message.modPow(e, n);
        BigInteger decrypted = encrypted.modPow(d, n);
        System.out.println("Prime numbers: p = " + p + ", q = " + q);
        System.out.println("Public key: (e, n) = (" + e + ", " + n + ")");
        System.out.println("Private key: (d, n) = (" + d + ", " + n + ")");
        System.out.println("Original message: " + message);
        System.out.println("Encrypted message: " + encrypted);
        System.out.println("Decrypted message: " + decrypted);}}
```

OUTPUT:-

```
Prime numbers: p = 61, q = 53
Public key: (e, n) = (7, 3233)
Private key: (d, n) = (1783, 3233)
Original message: 42
Encrypted message: 240
Decrypted message: 42
```

RESULT:-