EXPNO:4 **DATE:**

RSA

Aim: ToimplementanencryptionalgorithmusingRsa.

Algorithm:

- Step1:Selecttwolargeprimenumbers,pandq.● Step2:Calculatethe modulus, n = p * q.
- Step3:ComputeEuler'stotientfunction,φ(n)=(p-1)*(q-1).
- Step4:Chooseapublicexponent,e,suchthat1<e<φ(n)andgcd(e,φ(n)) =
  1.
- Step5:Computetheprivateexponent,d,suchthat(d*e)modφ(n)=1.
- Step6:Convertttheplaintextmessageintoanumericalrepresentation, usually using ASCII values or Unicode.
- Step7:Encryptthemessagebycomputingciphertext,c,usingtheformulac =(msg^e)modn.
- Step8:Printtheencrypteddata.
- Step9:Decryptthetheciphertextbycomputingtheoriginalmessage,m,using the formula m = (c^d) mod n.
- Step10:Printtheoriginalmessage.
- Step11:Return0forsuccessfulexecutionandprogramtermination.

Program:

```
import java.io.*;
importjava.math.*;
import java.util.*;
public class GFG {
        publicstaticdoublegcd(doublea,doubleh)
        {
                doubletemp;
```

```
                while(true){
        temp = a % h;
if(temp==0) return
h;
a=h;

h=temp;
                }
        }
publicstaticvoidmain(String[]args)
        {
                double p = 9;
                double q = 5;
                double n = p * q;
                double e = 2;
                doublephi=(p-1)*(q-1); while
                (e < phi) {
                        if(gcd(e,phi)==1)
                                break;
                        else
                              e++;
                }
                intk =2;
                doubled=(1+(k*phi))/ e;


                double msg = 12;

                System.out.println("Messagedata="+msg);


                doublec=Math.pow(msg,e); c
                 = c % n;
                System.out.println("Encrypteddata="+c);
                double m = Math.pow(c, d);
```

```
            m=m%n;
             System.out.println("OriginalMessageSent="+ m);
        }
}
```

Output:

```
java -cp /tmp/RgOMJoXiEh/GFG
Message data = 12.0
Encrypted data = 18.0
Original Message Sent = 29.0

=== Code Execution Successful ===
```

Result: