**Name: Kshitija Shamrao Jadhav**

**PRN: 2019BTECS00053**

**BATCH: B6**

**RSA:**

## **Algorithms**

Begin

   1. Choose two prime numbers p and q.

   2. Compute n = p\*q.

   3. Calculate phi = (p-1) \* (q-1).

   4. Choose an integer e such that 1 < e < phi(n) and gcd(e, phi(n)) = 1; i.e., e and phi(n) are coprime.

   5. Calculate d as d ≡ e−1 (mod phi(n)); here, d is the modular multiplicative inverse of e modulo phi(n).

   6. For encryption, c = me mod n, where m = original message.

   7. For decryption, m = c d mod n.

End

#include<iostream>

#include<math.h>

using namespace std;

// find gcd

int gcd(int a, int b) {

   int t;

   while(1) {

      t= a%b;

      if(t==0)

      return b;

      a = b;

      b= t;

   }

}

int main() {

   //2 random prime numbers

   double p = 389;

   double q = 397;

   double n=p\*q;//calculate n

   double track;

   double phi= (p-1)\*(q-1);//calculate phi

   //public key

   //e stands for encrypt

   double e=7;

   //for checking that 1 < e < phi(n) and gcd(e, phi(n)) = 1; i.e., e and phi(n) are coprime.

   while(e<phi) {

      track = gcd(e,phi);

      if(track==1)

         break;

      else

         e++;

   }

   //private key

   //d stands for decrypt

   //choosing d such that it satisfies d\*e = 1 mod phi

   double d1=1/e;

   double d=fmod(d1,phi);

   double message = 9;

   double c = pow(message,e); //encrypt the message

   double m = pow(c,d);

   c=fmod(c,n);

   m=fmod(m,n);

   cout<<"Original Message = "<<message;

   cout<<"\n"<<"p = "<<p;

   cout<<"\n"<<"q = "<<q;

   cout<<"\n"<<"n = pq = "<<n;

   cout<<"\n"<<"phi = "<<phi;

   cout<<"\n"<<"e = "<<e;

   cout<<"\n"<<"d = "<<d;

   cout<<"\n"<<"Encrypted message = "<<c;

   cout<<"\n"<<"Decrypted message = "<<m;

   return 0;

}

**Output:**

