Algoritmul RSA

1	**Functie** Algoritmul_RSA(p, q, e, m):
2	produs_pq <- p*q
3	phi <- (p-1)*(q-1)
4	coprim(e, phi)
5	mesaj_criptat <- cripteaza_mesaj(m, e, produs_pq)
6	mesaj_decriptat <-decripteaza_mesaj(m, e, phi, produs_pq)

1	**Functia** coprim(e, phi):
2	cat_timp e este mai mic decat phi executa:
3	daca numerele sunt coprime [cmmdc(e ,phi) = 1]: intrerupe
4	altfel: incrementeaza e [e <- e+1]
5	returneaza e

1	**Functia** cripteaza_mesaj(m, e, produs_pq):
2	mesaj_criptat <- [(mesaj ^ e) mod produs_pq]
3	returneaza mesaj_criptat

1	**Functia** decripteaza_mesaj(m, e, phi, produs_pq):
2	mesaj_decriptat <- [(mesaj ^ e) ^ (1/e mod phi)] mod produs_pq
3	returneaza mesaj_decriptat

CODUL++

```
#include <bits/stdc++.h>

using namespace std;

class RSA{

private:

   double numar_p, numar_q, mesaj, exponent;

public:

   RSA(double numar_p, double numar_q, double mesaj, double exponent);
```

```
   int cmmdc(int numar_a, int numar_b);
   double cripteaza_mesaj(double mesaj, double exponent, double produs_pq);
   double decripteaza_mesaj(double mesaj, double exponent, double phi, double produs_pq);
   double coprim(double exponent, double phi);
   void algoritm_RSA(double numar_p, double numar_q, double exponent, double mesaj);
};

// Initializare
RSA::RSA(double numar_p, double numar_q, double mesaj, double exponent){
      this->numar_p = numar_p;
      this->numar_q = numar_q;
      this->mesaj = mesaj;
      this->exponent = exponent;
}
// cel mai mare divizor comun
int RSA::cmmdc(int numar_a, int numar_b){
   return __gcd(numar_a,numar_b)==1;
}
// cripteaza mesaj
double RSA::cripteaza_mesaj(double mesaj, double exponent, double produs_pq){
   // mesaj_criptat = (mesaj ^ exponent) mod produs_pq
   double mesaj_criptat = fmod(pow(mesaj,exponent),produs_pq);
   return mesaj_criptat;
}
// decripteaza mesaj
double RSA::decripteaza_mesaj(double mesaj, double exponent, double phi, double produs_pq){
   // mesaj_decriptat =  ((mesaj ^ exponent)^ ((1/exponent) mod phi)) mod produs_pq
   double mesaj_decriptat = fmod(pow(pow(mesaj,exponent),fmod(1/exponent,phi)),produs_pq);
   return mesaj_decriptat;
}
// coprim
```

```
double RSA::coprim(double exponent, double phi){

   while(exponent < phi) {

     if(cmmdc(exponent,phi)==1)
        break;
     else
        exponent++;
  }
  return exponent;
}

void RSA::algoritm_RSA(double numar_p, double numar_q, double exponent, double mesaj){

  double produs_pq = numar_p*numar_q;
  double phi = (numar_p-1)*(numar_q-1);
  coprim(exponent, phi);
  double mesaj_criptat = cripteaza_mesaj(mesaj, exponent, produs_pq);
  double mesaj_decriptat = decripteaza_mesaj(mesaj, exponent, phi, produs_pq);

  cout<<"Mesaj original = "<<mesaj;
  cout<<"\n"<<"Mesaj criptat = "<<mesaj_criptat;
  cout<<"\n"<<"Mesaj decriptat = "<<mesaj_decriptat;

}

int main(){
   double numar_p = 13;
   double numar_q = 11;
   double exponent=7;
   double mesaj = 9;
```

```
    RSA* obj = new RSA(numar_p, numar_q, mesaj, exponent);
    obj->algoritm_RSA(numar_p, numar_q, exponent, mesaj);


    return 0;
}
```