Name: Rutikesh Sawant
Batch: B2
Subject: CNS Lab
PRN: 2019BTECS00034

Assignment 11

Aim: Implementation of RSA algorithm.

Theory:
The RSA algorithm is an asymmetric cryptography algorithm; this means that it uses a public key and a private key (i.e two different, mathematically linked keys). As their names suggest, a public key is shared publicly, while a private key is secret and must not be shared with anyone.
The RSA algorithm ensures that the keys, in the above illustration, are as secure as possible.

Code:

```
#include <bits/stdc++.h>
using namespace std;

void file()
{
#ifndef ONLINE_JUDGE
   freopen("input.txt", "r", stdin);
   freopen("output.txt", "w", stdout);
#endif
}

// Function for extended Euclidean Algorithm
int ansS, ansT;
int findGcdExtended(int r1, int r2, int s1, int s2, int t1, int t2)
{
   // Base Case
```

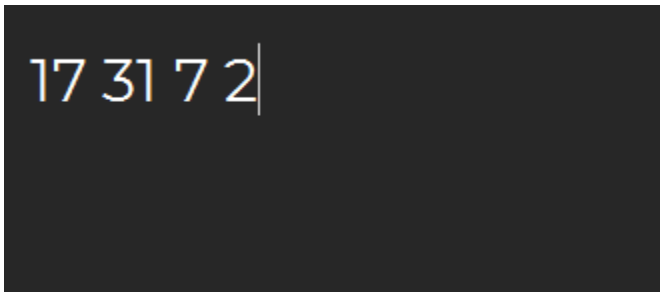
```
    if (r2 == 0)
    {
        ansS = s1;
        ansT = t1;
        return r1;
    }

    int q = r1 / r2;
    int r = r1 % r2;

    int s = s1 - q * s2;
    int t = t1 - q * t2;

    cout << q << " " << r1 << " " << r2 << " " << r << " " << s1 << " " << s2 << " " << s << "
" << t1 << " " << t2 << " " << t << endl;

    return findGcdExtended(r2, r, s2, s, t2, t);
}


int modInverse(int A, int M)
{
    int x, y;
    int g = findGcdExtended(A, M, 1, 0, 0, 1);
    if (g != 1) {
        cout << "Inverse doesn't exist";
        return 0;
    }
    else {

        // m is added to handle negative x


        int res = (ansS % M + M) % M;
```

```
        cout << "inverse is" << res << endl;
        return res;
    }
}

long long powM(long long a, long long b, long long n)
{
    if (b == 1)
        return a % n;
    long long x = powM(a, b / 2, n);
    x = (x * x) % n;
    if (b % 2)
        x = (x * a) % n;
    return x;
}

int findGCD(int num1, int num2)
{
    if (num1 == 0)
        return num2;
    return findGCD(num2 % num1, num1);
}

// Code to demonstrate RSA algorithm
int main()
{
    file();

    // Two random prime numbers
    long long p, q, e, msg;
    //17 31 7 2

    cout << "Please enter 2 prime number and e and Message to Encrypt" << endl;
    cin >> p >> q >> e >> msg;
```

```
cout << "2 random prime numbers selected are " << p << " " << q << endl;

// First part of public key:
long long n = p * q;
cout << "Product of two prime number n is " << n << endl;

// Finding other part of public key.
// e stands for encrypt

cout << "Taken e is " << e << endl;

long long phi = (p - 1) * (q - 1);
cout << "phi is " << phi << endl;

while (e < phi) {
    // e must be co-prime to phi and
    // smaller than phi.
    if (findGCD(e, phi) == 1)
        break;
    else
        e++;
}

cout << "Final e value is " << e << endl;



// Private key (d stands for decrypt)

long long d = modInverse(e, phi);
cout << "d is " << d << endl;

cout << "\nso now our public key is " << "<" << e << "," << n << ">" << endl;
cout << "\nso now our private key is " << "<" << d << "," << n << ">" << endl << endl;
```

```
    // Message to be encrypted

    cout << "Message date is " << msg << endl;

    // Encryption c = (msg ^ e) % n
    long long c = powM(msg, e, n);
    cout << "Encripted Message is " << c << endl;

    // Decryption m = (c ^ d) % n
    long long m = powM(c, d, n);
    cout << "original Message is " << m << endl;

    return 0;
}
```

Output:



```
Please enter 2 prime number and e and Message to Encrypt
2 random prime numbers selected are 17 31
Product of two prime number n is 527
Taken e is 7
phi is 480
Final e value is 7
0 7 480 7 1 0 1 0 1 0
68 480 7 4 0 1 -68 1 0 1
1 7 4 3 1 -68 69 0 1 -1
1 4 3 1 -68 69 -137 1 -1 2
3 3 1 0 69 -137 480 -1 2 -7
inverse is343
d is 343

so now our public key is <7,527>

so now our private key is <343,527>

Message date is 2
Encripted Message is 128
original Message is 2
```