**Project 5 - RSA Public Key Algorithm**

**1. Source code**

#include <iostream>

using namespace std;

// Function to find highest common factor

int hcf(int a, int b)

{

    return b == 0 ? a : hcf(b, a % b);

}

// Function to check if number is prime number in range [5000,10000]

bool checkPrime(int n)

{

    // check if in range of [5000,10000]

    if (n < 5000 || n > 10000)

    {

        return false;

    }

    else

    {

        for (int i = 2; i <= n / 2; i++)

        {

            if (n % i == 0)

            {

                return false;

            }

        }

        return true;

    }

}

// Function to find public key

int find\_public\_key(int totient\_function)

{

    int e;

    for (int i = 3; i++; i < totient\_function)

    {

        if (hcf(i, totient\_function) == 1)

        {

            e = i;

            break;

        }

    }

    return e;

}

// Function fo find private key

int find\_private\_key(int totient\_function, int e)

{

    bool signal = true;

    int d = int(totient\_function / e) + 1;

    while (signal)

    {

        if ((d \* e) % totient\_function == 1)

        {

            return d;

        }

        d += 1;

    }

    return -1;

}

// Function to check if d and e are multiplicative inverse

bool check\_multiplicative\_inverse(int d, int e, int totient\_function)

{

    if ((d \* e) % totient\_function == 1)

    {

        return true;

    }

    return false;

}

int main()

{

    int p;

    int q;

    cout << "Please enter a number p (prime number in range[5000,10000]): " << endl;

    cin >> p;

    cout << "Please enter a number q (prime number in range[5000,10000]): " << endl;

    cin >> q;

    while (!(checkPrime(p) && checkPrime(q)))

    {

        cout << "You entered incorrect values for p or q! Please enter again." << endl;

        cout << "Please enter a number p (prime number in range[5000,10000]): " << endl;

        cin >> p;

        cout << "Please enter a number q (prime number in range[5000,10000]): " << endl;

        cin >> q;

    }

    cout << "-------------------------------------------------" << endl;

    cout << "            Validation for p and q" << endl;

    cout << "p and q are prime numbers in range[5000,10000]" << endl;

    cout << "p: " << p << ", q: " << q << endl;

    int n = p \* q;

    cout << "-------------------------------------------------" << endl;

    cout << "Modulo is: " << n << endl;

    int totient\_function = (p - 1) \* (q - 1);

    cout << "Totient function is " << totient\_function << endl;

    int e = find\_public\_key(totient\_function);

    cout << "Public key is: <" << e << "," << n << ">" << endl;

    int d = find\_private\_key(totient\_function, e);

    cout << "Private key is: <" << d << "," << n << ">" << endl;

    cout << "-------------------------------------------------" << endl;

    cout << "        Check d and e are multplicative inverse" << endl;

    if (check\_multiplicative\_inverse(d, e, totient\_function))

    {

        cout << d << " and " << e << " are multiplicative inverse of each other based on " << totient\_function << endl;

    }

    else

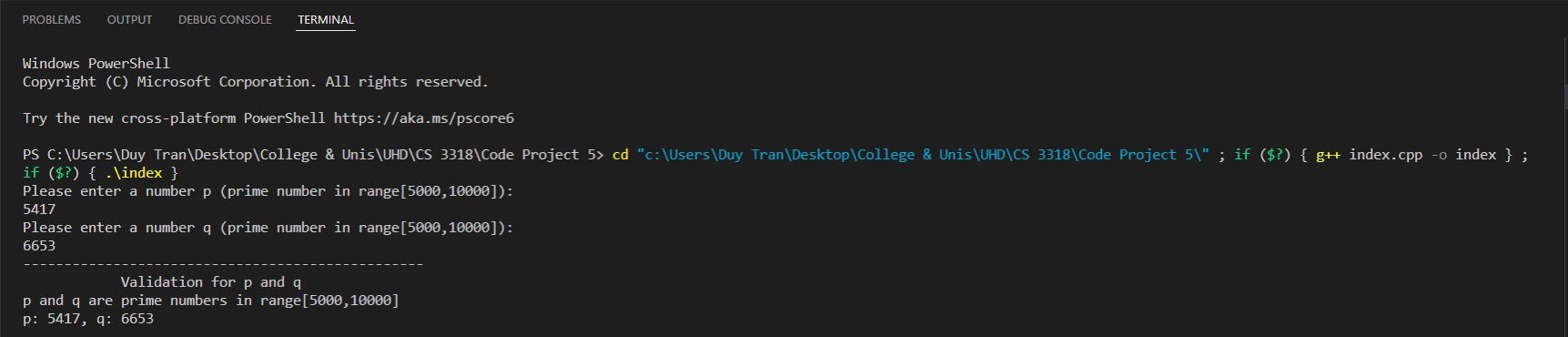
    {

        cout << d << " and " << e << " are not multiplicative inverse of each other based on " << totient\_function << endl;

    }

}

**2. A screen shot showing the validation and p, q**

****

**3. A screen shot showing n, O, a private key, and public key.**

****

**4. A screen shot showing the verification that d and e are multiplicative inverse for O**

****