**Cryptography and Network Security**

**Lab**

**Assignment No. 4**

**Name : Sourabh Shankar Patil**

**PRN : 21510045**

**Batch : B2**

**Chinese Remainder Theorem (CRT)**

The **Chinese Remainder Theorem** states that if you have a system of simultaneous congruences with pairwise coprime moduli, then there exists a unique solution modulo the product of the moduli.

#include <bits/stdc++.h>

using namespace std;

// Function to perform the Extended Euclidean Algorithm to find modular inverses

int extendedGCD(int a, int b, int &x, int &y)

{

    if (b == 0)

    {

        x = 1;

        y = 0;

        return a;

    }

    int x1, y1;

    int gcd = extendedGCD(b, a % b, x1, y1);

    x = y1;

    y = x1 - (a / b) \* y1;

    return gcd;

}

// Function to find the modular inverse of a under modulo m

int modInverse(int a, int m)

{

    int x, y;

    int g = extendedGCD(a, m, x, y);

    if (g != 1)

    {

        return -1; // Modular inverse doesn't exist

    }

    else

    {

        return (x % m + m) % m;

    }

}

// Function to apply the Chinese Remainder Theorem

int chineseRemainderTheorem(vector<int> num, vector<int> rem)

{

    int k = num.size(); // Number of equations

    int N = 1;

    // Compute product of all numbers

    for (int i = 0; i < k; i++)

    {

        N \*= num[i];

    }

    int result = 0;

    // Apply the formula x = sum(ai \* Ni \* Mi) % N

    for (int i = 0; i < k; i++)

    {

        int Ni = N / num[i];

        int Mi = modInverse(Ni, num[i]);

        result += rem[i] \* Ni \* Mi;

    }

    return result % N;

}

int main()

{

    vector<int> num = {3, 5, 7}; // Moduli

    vector<int> rem = {2, 3, 2}; // Remainders

    int result = chineseRemainderTheorem(num, rem);

    cout << "The solution x is: " << result << endl;

    return 0;

}

