**Name : Akash Babu Misal**

**PRN: 21520005**

**Class:** Final Year (Computer Science and Engineering)

**Course Name: Cryptography and Network Security**  **Lab**

**Assignment No – 11**

Chinese Remainder Theorem

**Aim**: Chinese Remainder Theorem implementation

**Theory**:

**Chinese Remainder Theorem Overview:**

The Chinese Remainder Theorem (CRT) is a mathematical theorem used in number theory and modular arithmetic. It provides a method for reconstructing an integer from its remainders when divided by several moduli. CRT is particularly useful in number theory and cryptography, where it allows for more efficient computations in modular arithmetic.

**The CRT Problem:**

Given a set of pairwise coprime moduli (m1, m2, ..., mk) and their respective remainders (a1, a2, ..., ak), CRT provides a solution for finding an integer 'x' that satisfies.

**Algorithm Steps:**

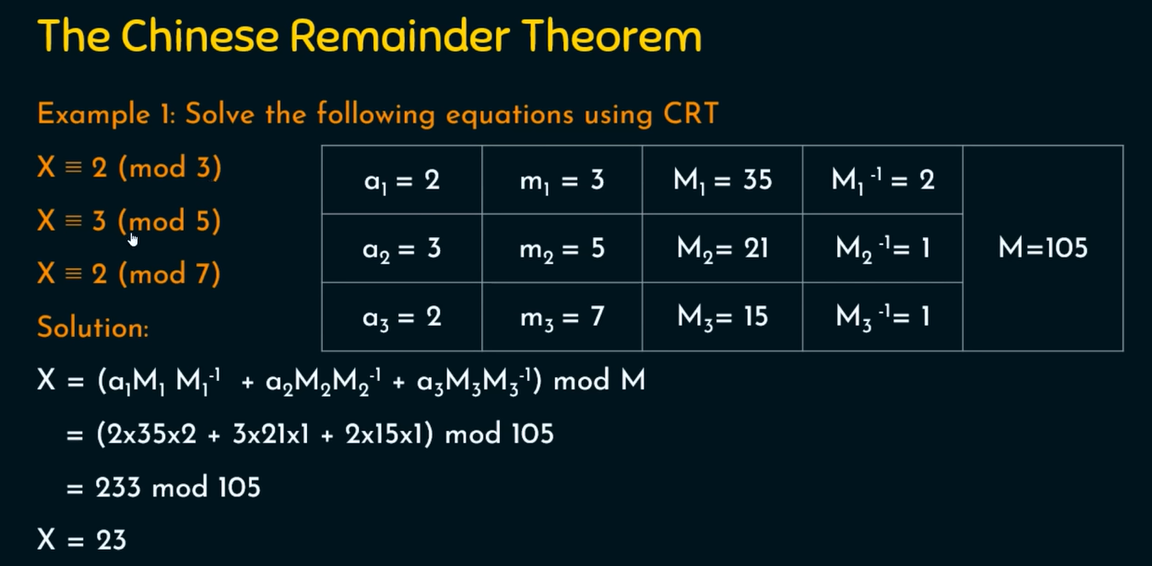
The steps to solve the system of congruences using the Chinese Remainder Theorem are as follows:

**Input**: Given a set of pairwise coprime moduli (m1, m2, ..., mk) and their corresponding remainders (a1, a2, ..., ak).

Calculate the Moduli Product (M): Compute M as the product of all the moduli, i.e., M = m1 \* m2 \* ... \* mk.

Calculate the Partial Products (Mi): For each modulus mi, calculate the partial product Mi as M divided by mi, i.e., Mi = M / mi.

Calculate the Inverse of Mi (Mi\_inverse): Find the modular multiplicative inverse of Mi modulo mi, denoted as Mi\_inverse. This step can be achieved using the Extended Euclidean Algorithm.



**Code:**

#include <iostream>

#include <bits/stdc++.h>

using namespace std;

long long find\_multiplicative\_inverse(long long a, long long b)

{

    long long q, r, t1 = 0, t2 = 1, t, main\_a = a;

    while (b > 0)

    {

        q = a / b;

        r = a % b;

        t = t1 - (t2 \* q);

        a = b;

        b = r;

        t1 = t2;

        t2 = t;

    }

    if (t1 < 0)

    {

        t1 += main\_a;

    }

    return t1;

}

int main()

{

    cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

    cout << "Chinese Remainder Theorem Problem  \n";

    cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

    cout << "Suppose that equation needs to be in form of X = a (mod m)\n";

    cout << "How many equations you want to perfrom : \t";

    int count;

    cin >> count;

    cout << "\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

    int M = 1;

    vector<int> a, m;

    for (int i = 0; i < count; i++)

    {

        cout << "Equation No : \t" << i + 1 << endl;

        cout << "Enter a :\t";

        int a\_data;

        cin >> a\_data;

        cout << "Enter m :\t";

        int m\_data;

        cin >> m\_data;

        a.push\_back(a\_data);

        m.push\_back(m\_data);

        cout << "\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

        M = M \* m\_data;

    }

    cout << "\nValue of M  :\t" << M << endl;

    vector<long long> M\_vector, M\_inverse\_vector;

    for (int i = 0; i < count; i++)

    {

        M\_vector.push\_back(M / m[i]);

    }

    for (int i = 0; i < count; i++)

    {

        M\_inverse\_vector.push\_back(find\_multiplicative\_inverse(m[i], M\_vector[i]));

    }

    long long sum = 0;

    for (int i = 0; i < count; i++)

    {

        sum += (a[i] \* M\_vector[i] \* M\_inverse\_vector[i]);

    }

    long long ans = sum % M;

    cout << "\nAfter calculations :\n";

    cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

    cout << "|\tEq. No\t|\ta[i]\t|\tm[i]\t|\tM[i]\t|\tM\_inverse[i]\t|\n";

    cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

    for (int i = 0; i < count; i++)

    {

        cout << "|\t" << i + 1 << "\t|\t" << a[i] << "\t|\t" << m[i] << "\t|\t" << M\_vector[i] << "\t|\t" << M\_inverse\_vector[i] << "\t\t|\n";

        cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

    }

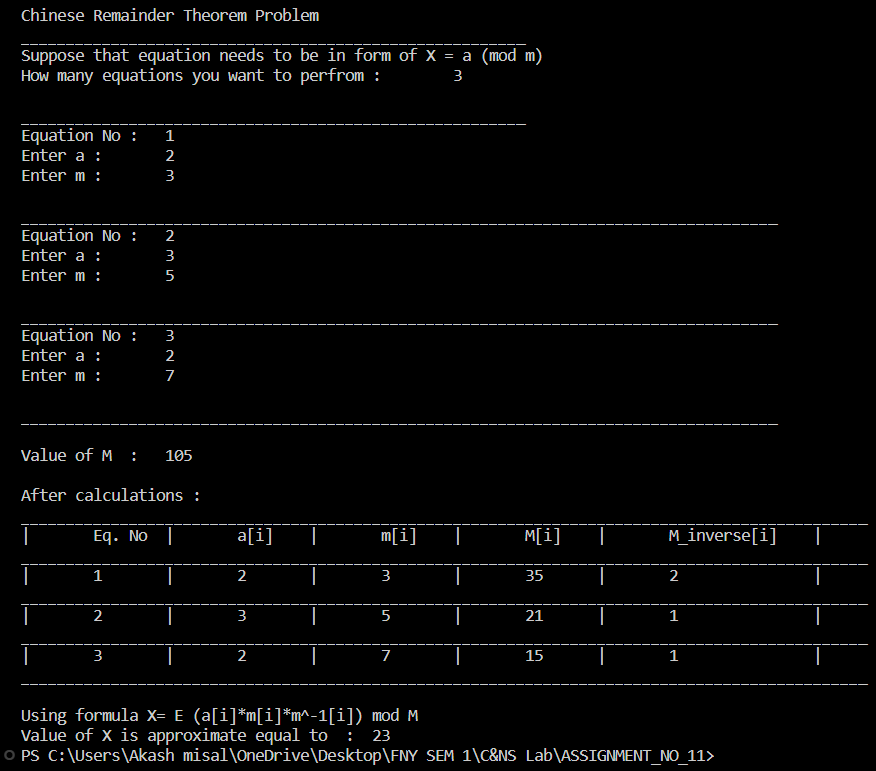
    cout << "\nUsing formula X= E (a[i]\*m[i]\*m^-1[i]) mod M \n";

    cout << "Value of X is approximate equal to  :  " << ans;

    return 0;

}

**Output:**

****

**Conclusion:**

The Chinese Remainder Theorem is a fundamental concept in number theory and modular arithmetic. This experiment allows us to understand and implement the CRT algorithm for solving a system of congruences efficiently. It demonstrates the power of CRT in reconstructing integers from remainders and its applications in various fields, including number theory, cryptography, and modular arithmetic.