**CNS LAB**

**Batch: B1**

**Assignment: 7**

**PRN No: 2020BTECS00006**

**Name: Samrat Vishwas Jadhav**

**Title of assignment:** Implementation of Chinese Remainder Theorem

1. **Aim:**

Implementation of Chinese Remainder Theorem

1. **Theory:**

Chinese Remainder Theorem:

If m1, m2, .., mk are pairwise relatively prime positive integers, and if a1, a2, .., ak are any integers, then the simultaneous congruences x ≡ a1 (mod m1), x ≡ a2 (mod m2), ..., x ≡ ak (mod mk) have a solution, and the solution is unique modulo m, where m = m1m2⋅⋅⋅mk .

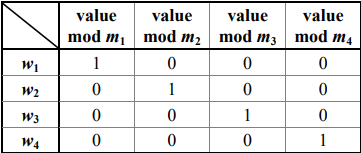
Proof that a solution exists:

To keep the notation simpler, we will assume k = 4. Note the proof is constructive, i.e., it shows us how to actually construct a solution.

Our simultaneous congruences are

x ≡ a1 (mod m1), x ≡ a2 (mod m2), x ≡ a3 (mod m3), x ≡ a4 (mod m4).

Our goal is to find integers w1, w2, w3, w4 such that:



Once we have found w1, w2, w3, w4, it is easy to construct x:

x = a1w1 + a2w2 + a3w3 + a4w4.

Moreover, as long as the moduli (m1, m2, m3, m4) remain the same, we can use the same w1, w2, w3, w4 with any a1, a2, a3, a4.

First define:

z1 = m / m1 = m2m3m4

z2 = m / m2 = m1m3m4

z3 = m / m3 = m1m2m4

z4 = m / m4 = m1m2m3

Note that

1. z1 ≡ 0 (mod mj) for j = 2, 3, 4.
2. ii) gcd(z1, m1) = 1.

(If a prime p dividing m1 also divides z1= m2m3m4, then p divides m2, m3, or m4.) and likewise for z2, z3, z4.

Next define:

y1 ≡ z1–1 (mod m1)

y2 ≡ z2 –1 (mod m2)

y3 ≡ z3 –1 (mod m3)

y4 ≡ z4 –1 (mod m4)

The inverses exist by (ii) above, and we can find them by Euclid’s extended algorithm.

Note that

1. y1z1 ≡ 0 (mod mj) for j = 2, 3, 4. (Recall z1 ≡ 0 (mod mj) )
2. iv) y1z1 ≡ 1 (mod m1) and likewise for y2z2, y3z3, y4z4.

Lastly define:

w1 ≡ y1z1 (mod m)

w2 ≡ y2z2 (mod m)

w3 ≡ y3z3 (mod m)

w4 ≡ y4z4 (mod m)

Then w1, w2, w3, and w4 have the properties in the above table.

#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;

*// cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";*

*// cout << "|\tQ\t|\tA\t|\tB\t|\tR\t|\tT1\t|\tT2\t|\tT\t|\n";*

*// cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";*

    while (b > 0)

    {

        q = a / b;

        r = a % b;

        t = t1 - (t2 \* q);

*// cout << "|\t" << q << "\t|\t" << a << "\t|\t" << b << "\t|\t" << r << "\t|\t" << t1 << "\t|\t" << t2 << "\t|\t" << t << "\t|\n";*

*// cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";*

        a = b;

        b = r;

        t1 = t2;

        t2 = t;

    }

*// cout << "|\t" << q << "\t|\t" << a << "\t|\t" << b << "\t|\t" << r << "\t|\t" << t1 << "\t|\t" << t2 << "\t|\t" << t << "\t|\n";*

*// cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";*

    if (t1 < 0)

    {

        t1 += main\_a;

    }

    return t1;

}

int main()

{

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

    cout << "Lets Solve 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]); *//caluculting M1,M2,M3*

    }

    for (int i = 0; i < count; i++)

    {

        M\_inverse\_vector.push\_back(find\_multiplicative\_inverse(m[i], M\_vector[i])); *//m1,m2,m3 and M1,M2,M3---M1\*M1^-1=1 mod m1;*

    }

    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|\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:**

