**Cryptography and Network Security Lab**

**Name: Sumit Narake**

**PRN: 2020BTECS00023**

**Batch: B2**

**Chinese Remainder Theorem**

**Aim:**

Implementation of Chinese Remainder Theorem.

**Theory:**

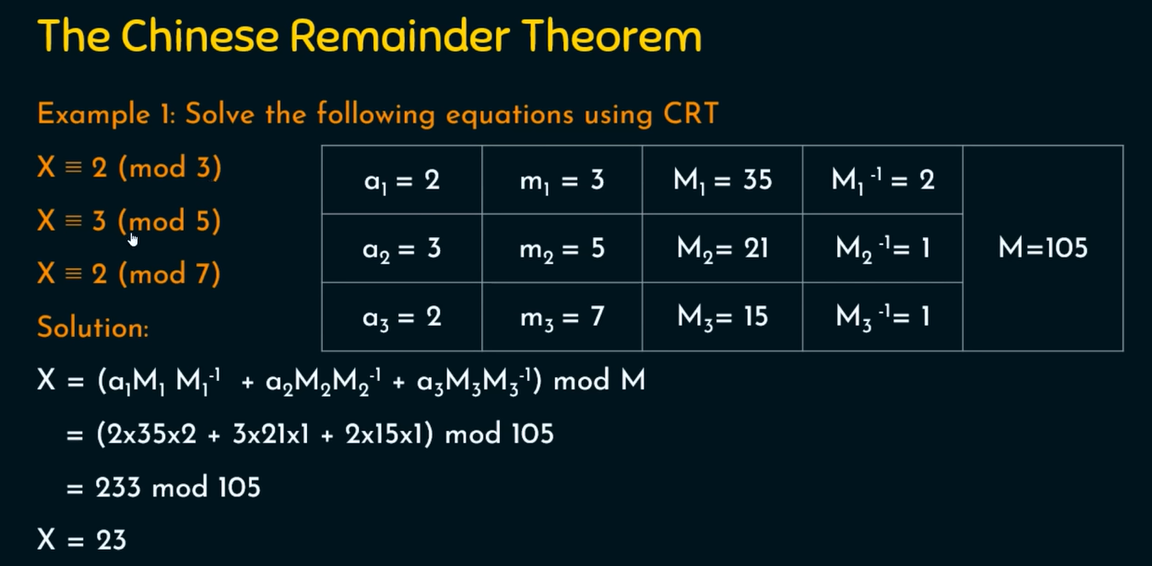
x = a1 (mod n1)

...

x = ak (mod nk)

This is equivalent to saying that x mod ni = ai (for i=1...k). The notation above is common in group theory, where you can define the group of integers modulo some number n and then you state equivalences (or congruence) within that group. So x is the unknown; instead of knowing x, we know the remainder of the division of x by a group of numbers. If the numbers ni are pairwise coprimes (i.e. each one is coprime with all the others) then the equations have exactly one solution. Such solution will be modulo N, with N equal to the product of all the ni.

Example:



**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:**

