**Experiment No.2**

**Aim:** Develop a program in C++ or Java based on number theory such as Chinese Remainder Theorem

**Objectives:** To study

* Chinese Remainder Theorem
* Set of residues
* Application of Chinese Remainder Theorem
* Modulo multiplicative inverse of a number.

**Theory:**

**Relative Prime Numbers**

Two integers are termed relative prime if the only common factor between them is 1.

i.e. Greatest Common Divisor(m, n) = 1

Any integer can be broken down into certain multiples of prime numbers. This is called prime factorization. When two integers are prime factorized and the only common number is 1, then the two integers are relative prime. Two distinct primes and are always relatively prime. Relative primality is not transitive.

Example

18 = 2 x 3 x 3

35 = 7 x 5

so 18 and 35 are relative primes.

18 = 2 x 3 x 3

21 = 3 x 7

3 is common, so 18 and 21 are not relative prime.

**Set of Residues**

It is a set of nonnegative integers less than n.

**Zn = {0,1,2,…….(n – 1) }**

**Chinese Remainder Theorem (CRT)**

Let m1, m2, …,mk be pair wise relatively prime positive integers. That is, gcd(mi, mj) = 1 for 1£i<j £k.

A can be computed as:



where for 1 ≤ i ≤ k.

**Steps in Chinese Remainder Theorem**

1. Find M = m1 × m2 × … × mk. This is the common modulus.

2. Find M1 = M/m1, M2 = M/m2… Mk = M/mk.

3. Find the multiplicative inverse of M1, M2, …, Mk using the corresponding moduli

(m1, m2, …,mk). Call the inverses M1−1, M2−1, …, Mk −1.

4. The solution to the simultaneous equations is



**Example1**

Represent 973 in Z1813 as a k-tuple.

**Answer:**

* M = 1813 = 37 \* 49 è m1 = 37 & m2 = 49
* A = 973
* A = (A mod m1, A mod m2) = (11, 42)

**Example2**

Find x for the following equations:

x≡2 mod 3

x≡ 3 mod 5

x≡ 2 mod 7

**Answer**

* + 1. M = 3 × 5 × 7 = 105
    2. M1 = 105 / 3 = 35, M2 = 105 / 5 = 21, M3 = 105 / 7 = 15
    3. The inverses are M1−1 = 2, M2−1 = 1, M3 −1 = 1
    4. x = (2 × 35 × 2 + 3 × 21 × 1 + 2 × 15 × 1) mod 105 = 23 mod 105
    5. x = 23

**INPUT:** Values of ai and mi

**OUTPUT:** Unique value of X

**Conclusion**: We have successfully implemented Chinese Remainder Theorem in C++.

**Source Code:**

#include<bits/stdc++.h>

using namespace std;

int inv(int a, int m)

{

int m0 = m, t, q;

int x0 = 0, x1 = 1;

if (m == 1)

return 0;

// Apply extended Euclid Algorithm

while (a > 1)

{

q = a / m; // q is quotient

t = m; // m is remainder now, process same as euclid's algo

m = a % m, a = t;

t = x0;

x0 = x1 - q \* x0;

x1 = t;

}

// Make x1 positive

if (x1 < 0)

x1 += m0;

return x1;

}

bool coprimality(int n[], int size)

{

for(int i = 0; i < size-1; i++)

{

for(int j = i+1; j < size; j++)

{

if(\_\_gcd(n[i], n[j]) != 1)

return false;

}

}

return true;

}

int main()

{

int size;

cout << "Enter number of inputs: ";

cin >> size;

int a[size], n[size], m[size], mi[size];

int M = 1, Y = 0;

cout <<"\nEnter in the format (a mod n) : \n";

for(int i = 0; i < size; i++)

{

cin >> a[i] >> n[i];

M \*= n[i];

}

// co-prime check

if(!coprimality(n, size))

{

cout << "Values of N aren't co-prime!";

return 0;

}

cout << "\nM : " << M << "\n";

for(int i = 0; i < size; i++)

{

m[i] = M / n[i];

mi[i] = inv(m[i], n[i]);

}

for(int i = 0; i < size; i++)

{

Y += (a[i] \* m[i] \* mi[i]);

}

cout << "X : " << Y % M;

}

**Snapshots :**

