**Government College of Engineering, Jalgaon**
**(An Autonomous Institute of Govt. of Maharashtra)**

| | | |
|---|---|---|
| **Name:** | | **PRN:** |
| **Class:** L.Y | **Semester:** VII | **Batch:** |
| **Date of Performance:** ______________ | | **Date of Completion:** _______________ |
| **Subject:** CO407U CNSL | | **Subject Teacher:** Ms. Shrutika Mahajan |

## PRATICAL NO. 5

**Aim :** Implementation of the Chinese Remainder Theorem (CRT) using Java programming.

**Theory :**

The **Chinese Remainder Theorem** provides a unique solution to a system of simultaneous linear congruences for a given set of **pairwise coprime moduli**.
A system of congruences:
$x \equiv a_1 \pmod{m_1}$
$x \equiv a_2 \pmod{m_2}$
...
$x \equiv a_n \pmod{m_n}$
where all moduli $m_1, m_2, \ldots, m_n$ are pairwise coprime ($\gcd(m_i, m_j) = 1$ for $i \neq j$), has a **unique solution**.

**Solution Formula:**

$x = ( \Sigma (a_i * M_i * y_i) ) \bmod M$

Where:

- $a_i$ = remainder for the i-th equation
- $m_i$ = modulus for the i-th equation
- M = product of all moduli: $M = m_1 * m_2 * \ldots * m_n$
- $M_i = M / m_i$
- $y_i$ = modular inverse of $M_i$ modulo $m_i$: $(M_i * y_i) \equiv 1 \pmod{m_i}$

**Note:** $y_i$ is computed using the **Extended Euclidean Algorithm**.

**General Case (Non-coprime moduli):**

A solution exists if each pair of congruences is compatible:
a_i ≡ a_j (mod gcd(n_i, n_j))
**Iterative method:**

1. Start with x = a1, m = n1.
2. For each next congruence x ≡ ai (mod ni):
   - Compute g = gcd(m, ni)
   - Check compatibility: (ai - x) % g == 0 → If not, no solution exists
   - Solve (m/g) * t ≡ (ai - x)/g (mod ni/g) using modular inverse
   - Update x = x + m * t and m = lcm(m, ni) = m * (ni/g)
   - Normalize x mod m
3. After all congruences, x is the solution modulo m.

**Algorithm:**

Step 1: Start
Step 2: Read k (number of congruences)
Step 3: For i = 1 to k:
    Read ai (remainder) and ni (modulus)
Step 4: Set x ← a1, m ← n1
Step 5: For i = 2 to k:
    g ← gcd(m, ni)
    If (ai - x) mod g ≠ 0 then
        No solution exists → Stop
    End If
    m1 ← m / g
    n1 ← ni / g
    rhs ← (ai - x) / g
    t ← (rhs * inverse(m1 mod n1)) mod n1
    x ← x + m * t
    m ← m * n1
    x ← x mod m
Step 6: Return x (solution) and m (modulus)
Step 7: Stop

**Program :**

```
import java.util.Scanner;

public class ChineseRemainderTheorem {
    public static int modInverse(int a, int m) {
        a = a % m;
        for (int x = 1; x < m; x++) {
            if ((a * x) % m == 1) {
                return x;
            }
        }
        return 1; // if not found, though for CRT we assume solution exists
    }
    public static int findX(int[] num, int[] rem, int k) {
        int prod = 1;
        for (int i = 0; i < k; i++) {
            prod *= num[i];
        }
        int result = 0;
        for (int i = 0; i < k; i++) {
            int pp = prod / num[i]; // partial product
            result += rem[i] * modInverse(pp, num[i]) * pp;
        }
        return result % prod;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter number of congruences: ");
        int k = sc.nextInt();

        int[] rem = new int[k];
        int[] num = new int[k];

        for (int i = 0; i < k; i++) {
            System.out.print("Enter remainder a" + (i + 1) + ": ");
            rem[i] = sc.nextInt();

            System.out.print("Enter modulus m" + (i + 1) + ": ");
            num[i] = sc.nextInt();
```

```
        }

        int x = findX(num, rem, k);

        int prod = 1;
        for (int i = 0; i < k; i++) {
            prod *= num[i];
        }
        System.out.println("\nx = " + x + " (mod " + prod + ")");
        System.out.println("Final Answer: x = " + x);

        sc.close();
    }
}
```

**Output :**

```
D:\Me\College\Practicals\Sem-7\CNSL>java ChineseRemainderTheorem
Enter number of congruences: 3
Enter remainder a1: 3
Enter modulus m1: 2
Enter remainder a2: 5
Enter modulus m2: 3
Enter remainder a3: 7
Enter modulus m3: 5

x = 17 (mod 30)
Final Answer: x = 17
```

**Conclusion :**

This program successfully implements the **Chinese Remainder Theorem** to find a unique solution for a system of linear congruences.

Key points:

- Relies on the **Extended Euclidean Algorithm** for modular inverses.
- Works for **pairwise coprime moduli**.
- Applications: Cryptography (e.g., RSA), large integer computations, and solving simultaneous congruences efficiently.

**Name and Sign of Teacher**

**Mrs. Shrutika Mahajan**