Walchand College of Engineering, Sangli (Government Aided Autonomous Institute)	
AY 2025-26	
Course Information	
Programme	B.Tech. (Computer Science and Engineering)
Class, Semester	Final Year B. Tech., Sem VII
Course Code	6CS451 C
Course Name	Cryptography and Network Security Lab
PRN	22510016

Experiment No. 04

Title - Implementation of Chinese Remainder Theorem (CRT)

Objectives:
To understand and implement the Chinese Remainder Theorem (CRT) for solving systems of simultaneous congruences with pairwise coprime moduli.

Problem Statement:
Given a system of simultaneous congruences:

$$x \equiv a1 (\mathrm{mod}\, n1)$$
$$x \equiv a2 (\mathrm{mod}\, n2)$$
$$\vdots$$
$$x \equiv ak (\mathrm{mod}\, nk)$$

where the moduli n1, n2,…,nk are pairwise coprime positive integers, find the smallest non-negative integer x that satisfies all these congruences simultaneously.

Equipment/Tools:

Computer or laptop
Python 3.x
Code editor/IDE

Theory:

The Chinese Remainder Theorem states that if n1, n2, …, nk are pairwise coprime, then the system of congruence's has a unique solution modulo N, where N = n1 * n2 * … * nk.
Let Mi = N/ni, and yi be the modular inverse of Mi modulo ni.
Then, the solution is:
x = (Σ ai * Mi * yi) mod N

Procedure:

Input the arrays of remainders (a) and moduli (n).
Compute N = product of all moduli.
For each congruence, compute Mi = N/ni.
Find yi = modular inverse of Mi mod ni using Extended Euclidean Algorithm.
Compute x = Σ (ai * Mi * yi).
Take x mod N to get the final answer.
Verify the result with all congruences.

Steps: Solvethe system:

$x \equiv 2 \pmod 3$
$x \equiv 3 \pmod 4$
$x \equiv 1 \pmod 5$

$N=60$

$M1=20 \rightarrow y1 = 2$
$M2=15 \rightarrow y2 = 3$
$M3=12 \rightarrow y3 = 3$

x=(2*20*2 + 3*15*3 + 1*12*3) mod 60 = 11

FinalAnswer: x = 11

```
from math import gcd
from functools import reduce

def egcd(a, b):
    if b == 0:
        return (a, 1, 0)
    g, x1, y1 = egcd(b, a % b)
    return(g,y1,x1    -(a//b)*     y1)

def modinv(a, m):
    g, x, _ = egcd(a, m)
    if g != 1:
        raise ValueError("Inverse does not exist")
    return x % m

def crt_pairwise_coprime(a, n):
    N=reduce(lambda   x,y:x*y,     n, 1)
    x = 0
    for ai, ni in zip(a, n):
        Mi=  N//ni
        yi = modinv(Mi, ni)
        x+=  ai*Mi   *yi
    return x % N, N
```

```python
print("Example 1:")
print("x ≡ 2 (mod 3), x ≡ 3 (mod 4), x ≡ 1 (mod 5)")
print("Solution:", crt_pairwise_coprime([2, 3, 1], [3, 4, 5]))
print()

print("Example 2:")
print("x ≡ 1 (mod 2), x ≡ 2 (mod 3), x ≡ 3 (mod 5)")
print("Solution:", crt_pairwise_coprime([1, 2, 3], [2, 3, 5]))
print()

print("Example 3:")
print("x ≡ 2 (mod 5), x ≡ 3 (mod 7), x ≡ 2 (mod 9)")
print("Solution:", crt_pairwise_coprime([2, 3, 2], [5, 7, 9]))
print()

print("Example 4:")
print("x ≡ 3 (mod 4), x ≡ 4 (mod 7), x ≡ 2 (mod 9)")
print("Solution:", crt_pairwise_coprime([3, 4, 2], [4, 7, 9]))
print()

print("Example 5:")
print("x ≡ 1 (mod 5), x ≡ 4 (mod 11), x ≡ 6 (mod 17)")
print("Solution:", crt_pairwise_coprime([1, 4, 6], [5, 11, 17]))
print()
```

```
PS C:\Users\aishw\OneDrive\Documents\Final Year\CNS Practicals\Assignment_4_23520001> python crt.py
Example 1:
x ≡ 2 (mod 3), x ≡ 3 (mod 4), x ≡ 1 (mod 5)
Solution: (11, 60)

Example 2:
x ≡ 1 (mod 2), x ≡ 2 (mod 3), x ≡ 3 (mod 5)
Solution: (23, 30)

Example 3:
x ≡ 2 (mod 5), x ≡ 3 (mod 7), x ≡ 2 (mod 9)
Solution: (227, 315)

Example 4:
x ≡ 3 (mod 4), x ≡ 4 (mod 7), x ≡ 2 (mod 9)
Solution: (11, 252)

Example 5:
x ≡ 1 (mod 5), x ≡ 4 (mod 11), x ≡ 6 (mod 17)
Solution: (686, 935)
```

Observations:

The solution is unique modulo N.
The Extended Euclidean Algorithm is efficient for finding modular inverses.
The method is much faster than brute force checking.

Conclusion:

The Chinese Remainder Theorem provides a systematic and efficient way to solve simultaneous congruences when moduli are pairwise coprime. The implementation works correctly and gives the smallest non-negative solution.