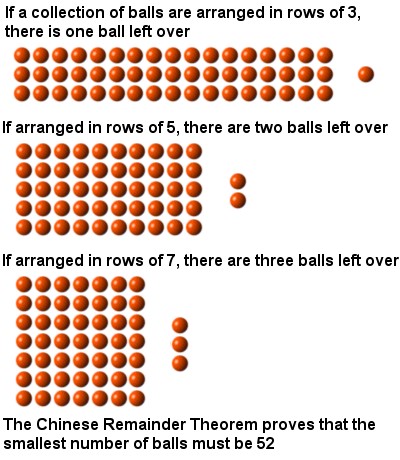
## Practical 10

**Aim:** Implement Chinese reminder theorem to a constraint satisfaction problem. Analyze its complexity.

**Theory:**

1. The Chinese remainder theorem is a theorem which gives a unique solution to simultaneous linear congruences with coprime moduli. In its basic form, the Chinese remainder theorem will determine a number *p* that, when divided by some given divisors, leaves given remainders.
2. In number theory, the Chinese remainder theorem states that if one knows the remainders of the Euclidean division of an integer n by several integers, then one can determine uniquely the remainder of the division of n by the product of these integers, under the condition that the divisors are pairwise coprime.

**Example:**



**Algorithm:**

def ChineseRemainderGauss(n, N, a):

result = 0

for i in range(len(n)):

ai = a[i] ni = n[i] bi = N // ni

result += ai \* bi \* invmod(bi, ni)

return result % N

**Code:**

def inv(a, m) :

m0 = m

x0 = 0

x1 = 1

if (m == 1) :

return 0

# Apply extended Euclid Algorithm

while (a > 1) :

# q is quotient

q = a // m

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 = x1 + m0

return x1

def findMinX(num, rem, k) :

# Compute product of all numbers

prod = 1

for i in range(0, k) :

prod = prod \* num[i]

# Initialize result

result = 0

# Apply above formula

for i in range(0,k):

pp = prod // num[i]

result = result + rem[i] \* inv(pp, num[i]) \* pp

return result % prod

# Driver method

num = [3, 4, 5]

rem = [2, 3, 1]

k = len(num)

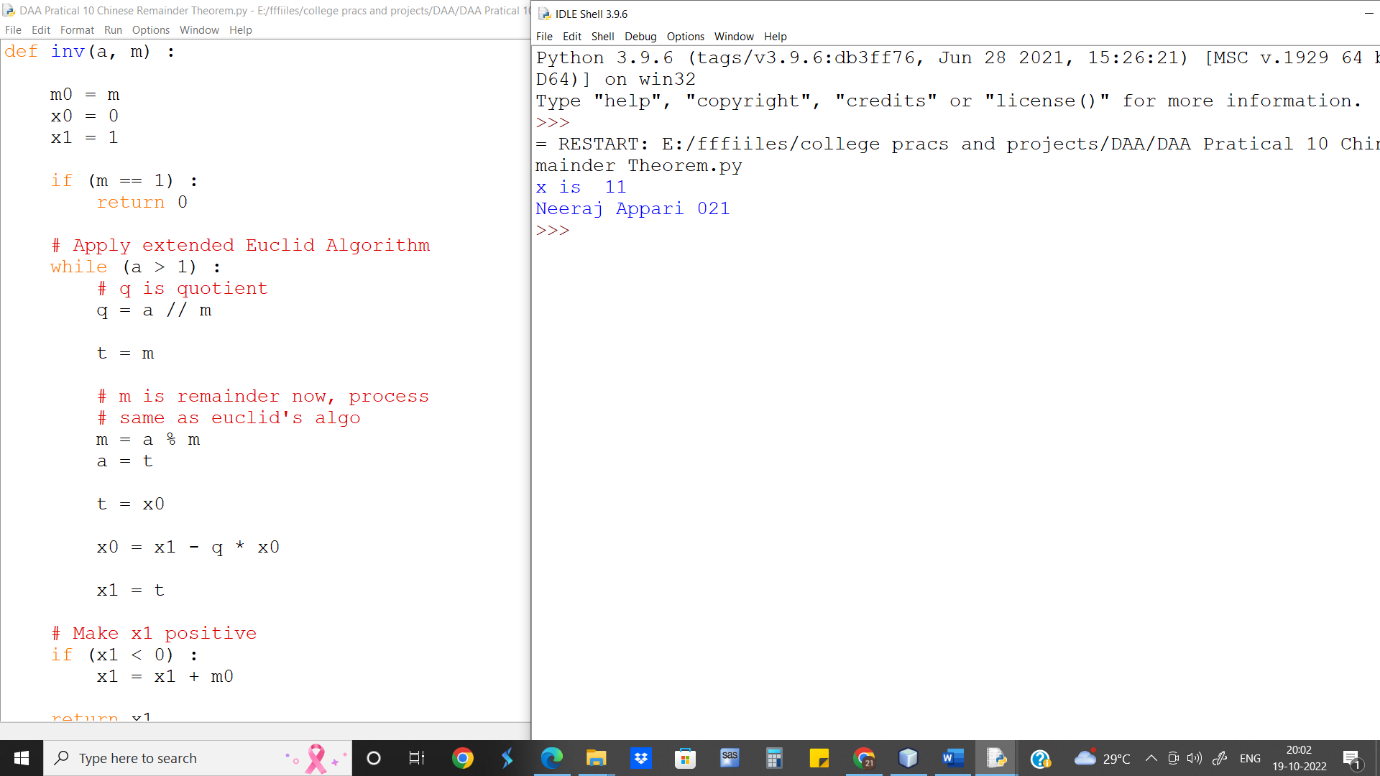
print( "x is " , findMinX(num, rem, k))

print("Neeraj Appari 021")

**Output**:

x is 11

Neeraj Appari 021



**Complexity Analysis:-** The complexity of Chinese reminder theorem is O(n) where n is the number of input / size of array.

The program only needs one loop to calculate the product of all the n.

**Conclusion:-** We implement Chinese reminder theorem and find that its complexity is O(n).