CRYPTOGRAPHY AND NETWORK SECURITY

LAB 3 : WRITE THE CODE AND EXECUTE FOR THE FOLLOWING ALGORITHMS:

- → 1. CHINESE REMAINDER THEOREM
- → 2. EXTENDED EUCLIDEAN ALGORITHM

NAME : OM SUBRATO DEY

REGISTER NO.: 21BAI1876

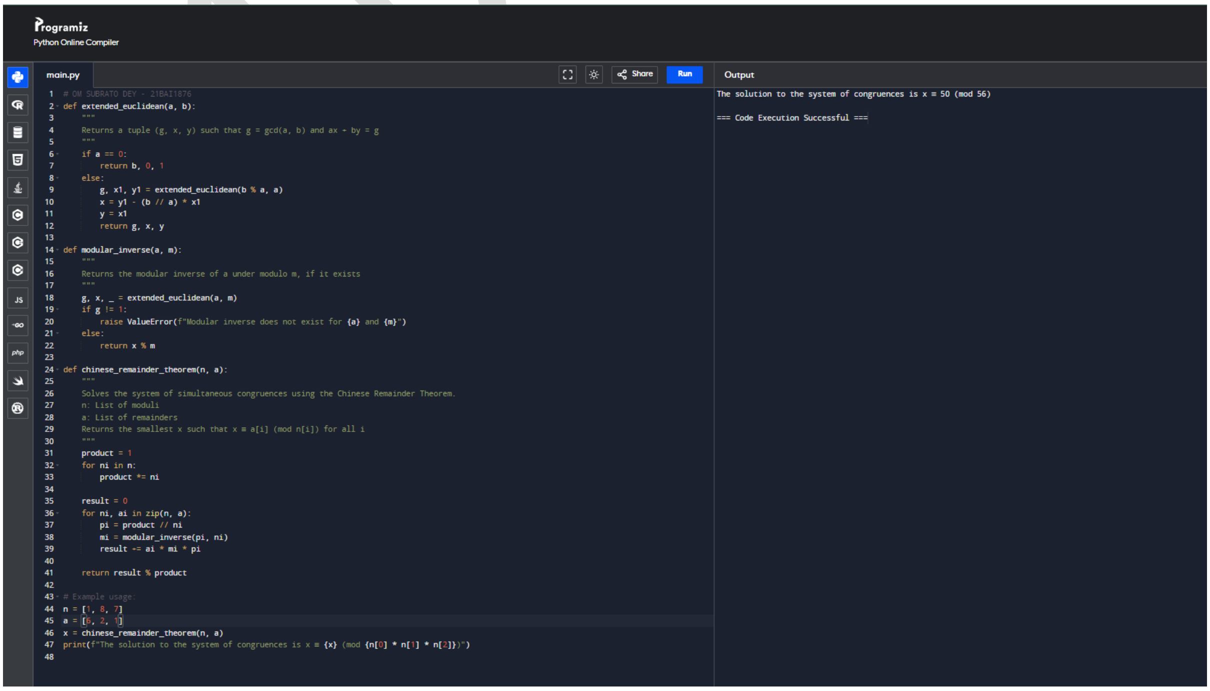
1. CHINESE REMAINDER THEOREM:

CODE:

```
def extended_euclidean(a, b):
    """
    Returns a tuple (g, x, y) such that g = gcd(a, b) and ax + by = g
    """
    if a == 0:
        return b, 0, 1
    else:
        g, x1, y1 = extended_euclidean(b % a, a)
        x = y1 - (b // a) * x1
        y = x1
        return g, x, y

def modular_inverse(a, m):
    """
    Returns the modular inverse of a under modulo m, if it exists
    """
    g, x, _ = extended_euclidean(a, m)
    if g != 1:
        raise ValueError(f"Modular inverse does not exist for {a} and {m}")
    else:
        return x % m

def chinese_remainder_theorem(n, a):
    """
    Solves the system of simultaneous congruences using the Chinese Remainder Theorem.
    n: List of moduli
    a: List of remainders
```

```
    Returns the smallest x such that x ≡ a[i] (mod n[i]) for all i
    """
    product = 1
    for ni in n:
        product *= ni
    result = 0
    for ni, ai in zip(n, a):
        pi = product // ni
        mi = modular_inverse(pi, ni)
        result += ai * mi * pi

    return result % product
# Example usage:
n = [1, 8, 7]
a = [6, 2, 1]
x = chinese_remainder_theorem(n, a)
print(f"The solution to the system of congruences is x ≡ {x} (mod {n[0] * n[1] * n[2]})")
```

OUTPUT:

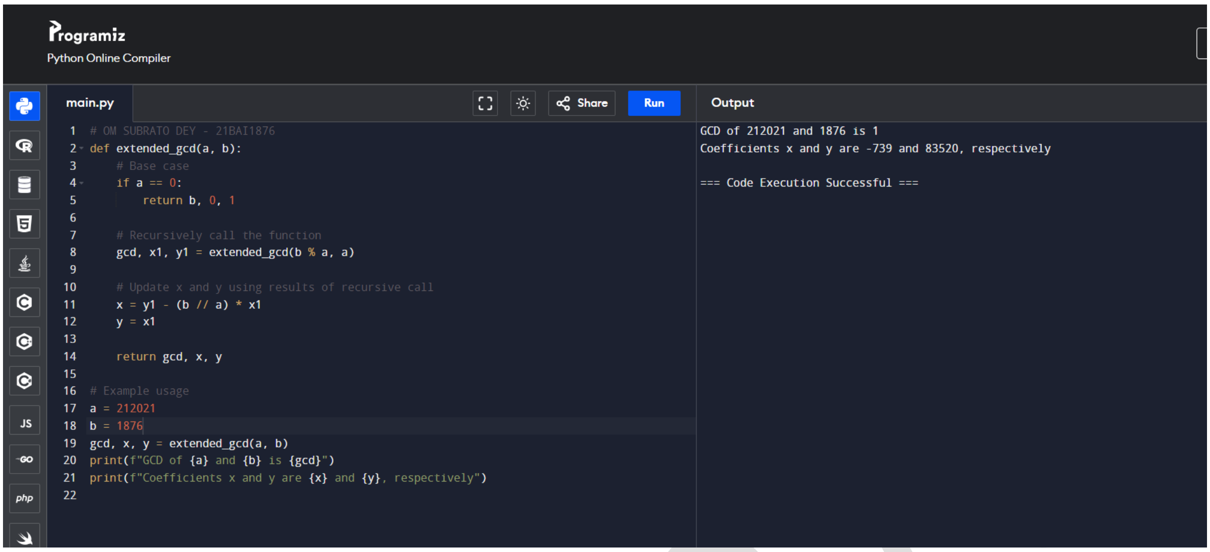


2. EXTENDED EUCLIDEAN ALGORITHM:

CODE:

```
# OM SUBRATO DEY - 21BAI1876
def extended_gcd(a, b):
    # Base case
    if a == 0:
        return b, 0, 1

    # Recursively call the function
    gcd, x1, y1 = extended_gcd(b % a, a)

    # Update x and y using results of recursive call
    x = y1 - (b // a) * x1
    y = x1

    return gcd, x, y

# Example usage
a = 212021
b = 1876
gcd, x, y = extended_gcd(a, b)
print(f"GCD of {a} and {b} is {gcd}")
print(f"Coefficients x and y are {x} and {y}, respectively")
```

OUTPUT:


Programiz
Python Online Compiler
main.py
Share
Run
# OM SUBRATO DEY - 21BAI1876
def extended_gcd(a, b):
# Base case
if a == 0:
return b, 0, 1
Recursively call the function
gcd, x1, y1 = extended_gcd(b % a, a)
# Update x and y using results of recursive call
x = y1 - (b // a) * x1
y = x1
return gcd, x, y
# Example usage
a = 212021
b = 1876
gcd, x, y = extended_gcd(a, b)
print(f"GCD of {a} and {b} is {gcd}")
print(f"Coefficients x and y are {x} and {y}, respectively")
Output
GCD of 212021 and 1876 is 1
Coefficients x and y are -739 and 83520, respectively
=== Code Execution Successful ===