Name:-Amrit Ranjan

Reg No:-21BCE1847

Cryptography And Network Security

Assignment 9

Faculty Name:-Dr Tapabarta Roy

Gcd of two natural numbers using Eucledian algorithm



Ques:-

Write down a program for determining the gcd of two natural numbers using Eucledian algorithm. Then implement Extended Eucledean algorithm to express the gcd as a linear sum of given integers. Use this to find inverse of a number modulo a given number (if exists).

Code:-

def gcd(a, b):

    while b:

        a, b = b, a % b

    return a

def extended\_gcd(a, b):

    if a == 0:

        return (b, 0, 1)

    else:

        gcd, x, y = extended\_gcd(b % a, a)

        return (gcd, y - (b // a) \* x, x)

def find\_inverse(a, m):

    gcd, x, y = extended\_gcd(a, m)

    if gcd != 1:

        raise ValueError(f"The inverse of {a} modulo {m} does not exist.")

    else:

        return x % m

def express\_as\_linear\_sum(numbers):

    n = len(numbers)

    if n == 0:

        return "No numbers provided"

    elif n == 1:

        return numbers[0], (1,)

    elif n == 2:

        gcd\_val = gcd(numbers[0], numbers[1])

        x, y = extended\_gcd(numbers[0], numbers[1])[1:]

        return gcd\_val, (x, y)

    else:

        gcd\_val, \*coefficients = express\_as\_linear\_sum(numbers[:-1])

        last\_number = numbers[-1]

        new\_gcd, x, y = extended\_gcd(gcd\_val, last\_number)

        new\_coefficients = [c \* x for c in coefficients] + [y]

        return new\_gcd, tuple(new\_coefficients)

def main():

    num1 = int(input("Enter the first number: "))

    num2 = int(input("Enter the second number: "))

    # Finding gcd

    gcd\_result = gcd(num1, num2)

    print("GCD of", num1, "and", num2, ":", gcd\_result)

    # Express gcd as a linear sum

    gcd\_linear\_sum = express\_as\_linear\_sum([num1, num2])

    print("GCD expressed as a linear sum:", gcd\_linear\_sum)

    # Finding inverse modulo

    num = int(input("Enter the number to find its inverse: "))

    modulo = int(input("Enter the modulo: "))

    try:

        inverse = find\_inverse(num, modulo)

        print("Inverse of", num, "modulo", modulo, ":", inverse)

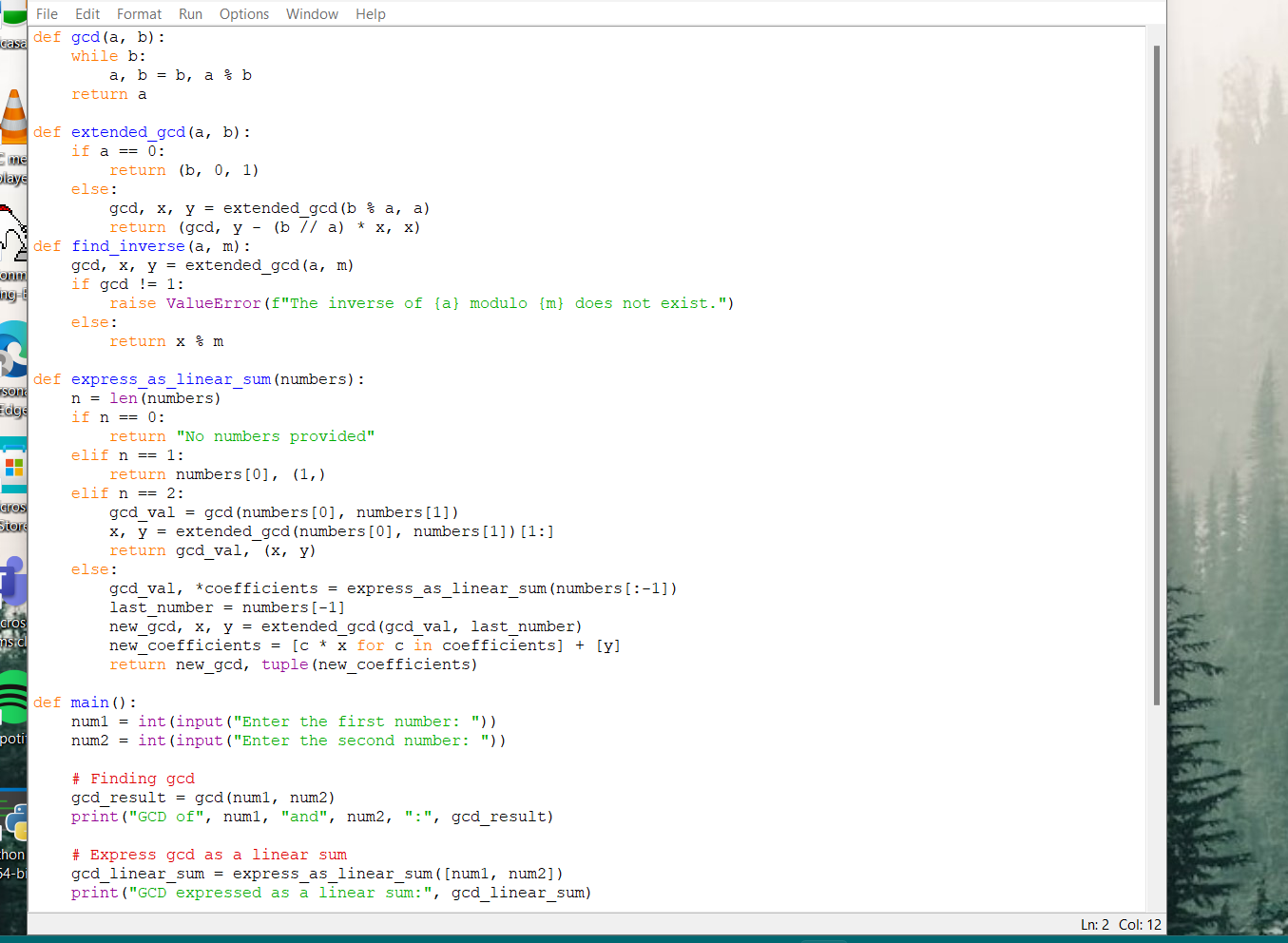
    except ValueError as e:

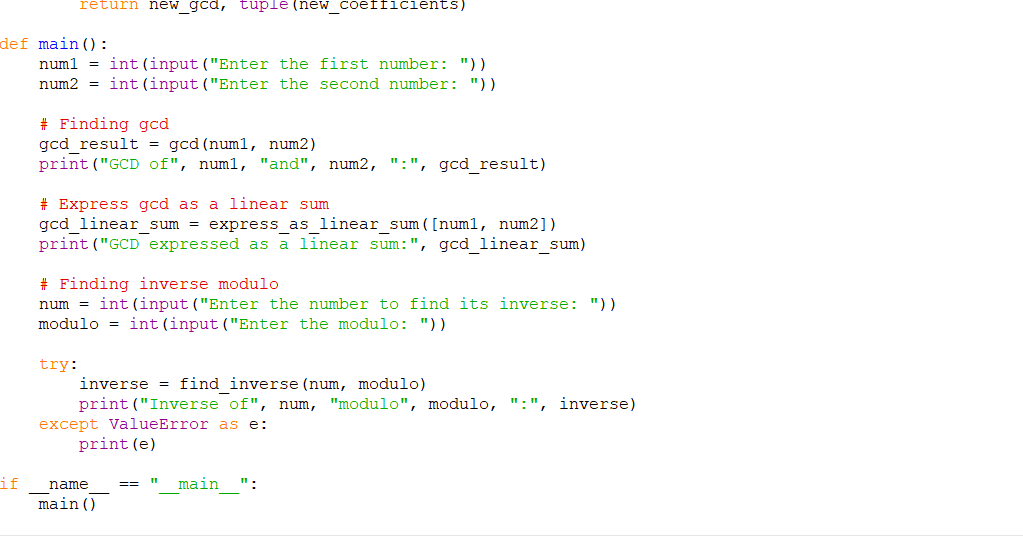
        print(e)

if \_\_name\_\_ == "\_\_main\_\_":

    main()

Code ScreenShots:-





Code Output Screenshots:-

