21BDS0340

Abhinav Dinesh Srivatsa

Cryptography and Network Security Lab

BCSE309P

Assessment – II

**Question 1**

Write a program for a four function calculator in GF(24)

**Aim**

To implement a four function calculator in GF(24)

**Algorithm**

1. For addition, do simple addition and apply the modulus after
2. For subtraction, do simple subtraction and apply the modulus after
3. For multiplication, do simple multiplication and apply the modulus after
4. For division, find the multiplicative inverse of the divisor and then multiply it with the dividend

**Code**

def add(x, y, mod):

return (x + y) % mod

def sub(x, y, mod):

return (x - y) % mod

def mult(x, y, mod):

return (x \* y) % mod

def inverse(x, mod):

return (x \*\* (mod - 2)) % mod

def div(x, y, mod):

return mult(x, inverse(y, mod), mod)

mod = 2\*\*4

m = int(input("enter number 1: "))

n = int(input("enter number 2: "))

print()

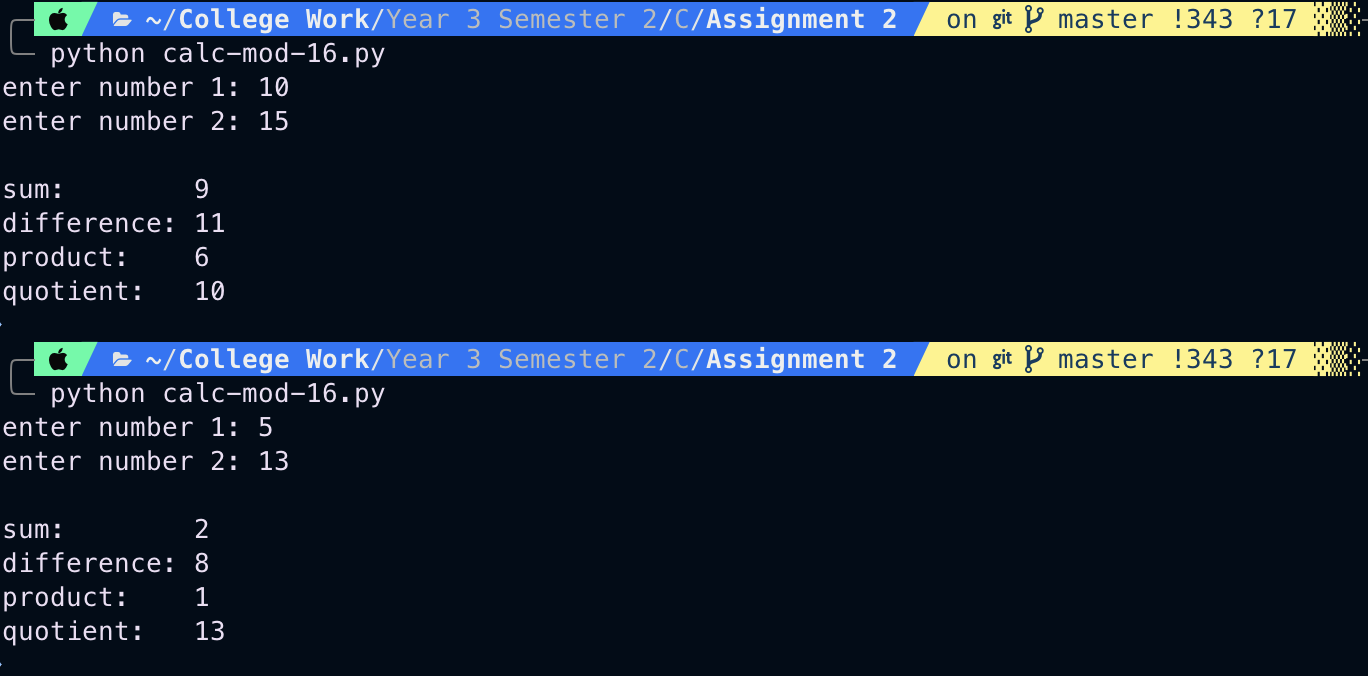
print(f"sum: {add(m, n, mod)}")

print(f"difference: {sub(m, n, mod)}")

print(f"product: {mult(m, n, mod)}")

print(f"quotient: {div(m, n, mod)}")

**Input/Output**



**Question 2**

Write a program that implements fast exponentiation modulo n

**Aim**

To implement fast exponentiation modulo n

**Algorithm**

1. Convert the exponent to binary, this helps in finding out the powers that matter to multiply
2. For each binary value, multiply the result and the base with it, then mod n
3. Multiply all the results together for the final answer

**Code**

def binary\_list(n):

binary = []

while n > 0:

binary.append(int(n % 2))

n = int(n / 2)

return binary

def exponentiate(m, n, mod):

binary\_power = binary\_list(n)

current\_val = m % mod

prod = 1

for i in binary\_power:

if i == 1:

prod = (prod \* current\_val) % mod

current\_val = (current\_val \* current\_val) % mod

return prod % mod

m = int(input("enter base: "))

n = int(input("enter power: "))

mod = int(input("enter mod: "))

print(f"{m}^{n} mod {mod} is {exponentiate(m, n, mod)}")

**Input/Output**

A computer screen with blue and white text

Description automatically generated

**Question 3**

Write a program that implements P-box where the permutation is defined by a table

**Aim**

To implement P-box given a table and code

**Algorithm**

1. Take input for P-box and the code
2. Permute the code based on the values in P-box
3. Return the output as the permuted code

**Code**

def apply(pbox, code):

output = []

for i in pbox:

output.append(code[i - 1])

return output

pbox\_raw = input("enter p-box substituition array: ").split()

pbox\_joined = ",".join(pbox\_raw)

pbox = eval(f"[{pbox\_joined}]")

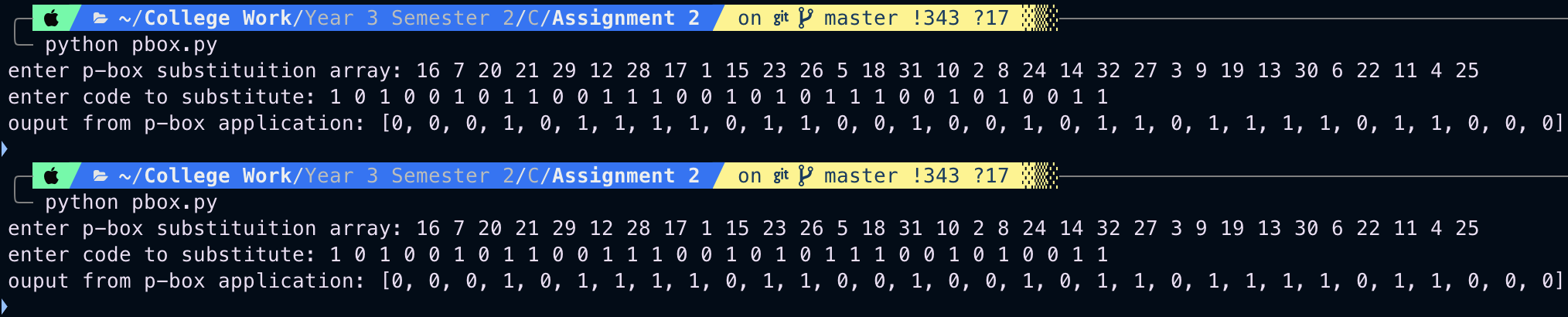
code\_raw = input("enter code to substitute: ").split()

code\_joined = ",".join(code\_raw)

code = eval(f"[{code\_joined}]")

print(f"ouput from p-box application: {apply(pbox, code)}")

**Input/Output**



**Question 4**

Write a program that implements S-box where the substitution is defined by a table

**Aim**

To implement S-box given a table and code

**Algorithm**

1. Take input for S-box and the code
2. Substitute the code based on the values in S-box
3. Return the output as the permuted code

**Code**

def apply(sbox, code):

row = code[0] \* 2 + code[5]

col = code[1] \* 8 + code[2] \* 4 + code[3] \* 2 + code[4]

sub\_val = sbox[row][col]

return binary(sub\_val)

def binary(num):

return bin(num)[2:].zfill(4)

sbox = [

[14, 4, 13, 1, 2, 15, 11, 8, 3, 10, 6, 12, 5, 9, 0, 7],

[0, 15, 7, 4, 14, 2, 13, 1, 10, 6, 12, 11, 9, 5, 3, 8],

[4, 1, 14, 8, 13, 6, 2, 11, 15, 12, 9, 7, 3, 10, 5, 0],

[15, 12, 8, 2, 4, 9, 1, 7, 5, 11, 3, 14, 10, 0, 6, 13]

]

code\_raw = input("enter code to substitute: ").split()

code\_joined = ",".join(code\_raw)

code = eval(f"[{code\_joined}]")

print(f"ouput from s-box application: {apply(sbox, code)}")

**Input/Output**

