1)

Find the LCM and GCD of n numbers?

Sample Input:

N value = 2

Number 1 = 16

Number 2 = 20

Sample Output:

LCM = 80

GCD = 4

Test cases:

a) N = 3, {12, 25, 30}

b) N = 2, {52, 25, }

c) N = 3, {17, 19, 11}

d) N = -2, {52, 60}

e) N = 2, {30, 45}

code:  
import math

def calculate\_lcm(numbers):

lcm = numbers[0]

for num in numbers[1:]:

lcm = lcm \* num // math.gcd(lcm, num)

return lcm

def calculate\_gcd(numbers):

gcd = numbers[0]

for num in numbers[1:]:

gcd = math.gcd(gcd, num)

return gcd

def find\_lcm\_gcd():

try:

N = int(input("Enter the value of N: "))

if N <= 0:

print("N should be a positive integer.")

return

numbers = []

for i in range(1, N + 1):

num = int(input(f"Enter Number {i}: "))

numbers.append(num)

lcm = calculate\_lcm(numbers)

gcd = calculate\_gcd(numbers)

print(f"\nLCM = {lcm}")

print(f"GCD = {gcd}")

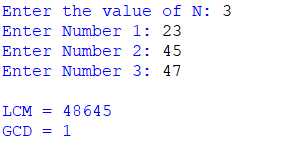
except ValueError:

print("Invalid input. Please enter valid integers.")

# Call the function to execute

find\_lcm\_gcd()

output:



2) Write a program to convert Decimal number equivalent to Binary number and octal numbers?

Sample Input:

Decimal Number: 15

Sample Outp

ut:

Binary Number = 1111

Octal = 17

Test cases:

a) 111

b) 15.2

c) 0

d) B12

e) 1A.2

code:

def decimal\_to\_binary(decimal\_num):

if decimal\_num < 0 or not isinstance(decimal\_num, int):

return "Invalid input. Please enter a positive integer."

binary\_num = bin(decimal\_num).replace("0b", "")

return binary\_num

def decimal\_to\_octal(decimal\_num):

if decimal\_num < 0 or not isinstance(decimal\_num, int):

return "Invalid input. Please enter a positive integer."

octal\_num = oct(decimal\_num).replace("0o", "")

return octal\_num

def convert\_decimal():

try:

decimal\_num = int(input("Enter Decimal Number: "))

binary\_num = decimal\_to\_binary(decimal\_num)

octal\_num = decimal\_to\_octal(decimal\_num)

print(f"\nBinary Number = {binary\_num}")

print(f"Octal = {octal\_num}")

except ValueError:

print("Invalid input. Please enter a valid integer.")

# Call the function to execute

convert\_decimal()

output:



3) Print the pattern

2

2 4

2 4 6

2 4 6 8

2 4 6 8 10

Code:

def print\_pattern(rows):

for i in range(1, rows + 1):

for j in range(1, i + 1):

print(2 \* j, end=" ")

print()

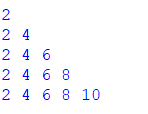
# Sample Input

rows = 5

# Sample Output

print\_pattern(rows)

Output:



14

Write a Python Program to remove duplicates from the sorted array

Sample Input: Array = {15, 14, 25, 14, 32, 14, 31} Sample Output: Sorted Array = {14, 15, 25, 31, 32}

Test cases:

1. {16, 16, 16 16, 16}

2. {0, 0, 0, 0}

3. {-12, -78, -35, -42}

4. {1,2,3,7,8,9,4,5,6}

5. {1-2,2-3,3-4,4-5,5-6"’.0.z

def remove\_duplicates(arr):

if not arr:

return []

# Initialize variables

result = [arr[0]]

# Traverse the sorted array and remove duplicates

for i in range(1, len(arr)):

if arr[i] != arr[i - 1]:

result.append(arr[i])

return result

# Sample Input

array = [15, 14, 25, 14, 32, 14, 31]

# Sorting the array

array.sort()

# Removing duplicates

result\_array = remove\_duplicates(array)

# Sample Output

print("Sorted Array without duplicates:", result\_array)

code:

def remove\_duplicates(arr):

if not arr:

return []

# Initialize variables

result = [arr[0]]

# Traverse the sorted array and remove duplicates

for i in range(1, len(arr)):

if arr[i] != arr[i - 1]:

result.append(arr[i])

return result

# Sample Input

array = [15, 14, 25, 14, 32, 14, 31]

# Sorting the array

array.sort()

# Removing duplicates

result\_array = remove\_duplicates(array)

# Sample Output

print("Sorted Array without duplicates:", result\_array)

output:



15)

Write a program to find the number of special characters in the given statement

Sample Input: Given statement: Modi Birthday @ September 17, #&$% is the wishes code for him.

Sample Output:Number of special Characters: 5

Code:  
def count\_special\_characters(statement):

special\_characters = 0

# Define special characters set

special\_chars = set('!@#$%^&\*()\_+-=[]{}|;\':",./<>?')

# Count special characters

for char in statement:

if char in special\_chars:

special\_characters += 1

return special\_characters

# Sample Input

given\_statement = "Modi Birthday @ September 17, #&$% is the wishes code for him."

# Count special characters

special\_count = count\_special\_characters(given\_statement)

# Sample Output

print("Number of special Characters:", special\_count)

Output:  


16

Sample Input:

Enter the numbers 4 , 54 29 71 7 59 98 23

Sample Output:

Composite number:3 Prime number:5

Test cases:

1. 33, 41, 52, 61,73,90

2. TEN, FIFTY, SIXTY-ONE, SEVENTY-SEVEN, NINE

3. 45, 87, 09, 5.0 ,2.3, 0.4

4. -54, -76, -97, -23, -33, -98

5. 45, 73, 00, 50, 67, 44"

Code:

def is\_prime(num):

if num <= 1:

return False

if num == 2:

return True # 2 is a prime number

if num % 2 == 0:

return False # other even numbers are not prime

# Check for odd factors up to the square root of num

for i in range(3, int(num\*\*0.5) + 1, 2):

if num % i == 0:

return False

return True

def count\_numbers(numbers):

composite\_count = 0

prime\_count = 0

# Process each number in the input

for num in numbers:

num = int(num) # Convert string to integer

if num <= 1:

continue # Skip numbers less than or equal to 1

if is\_prime(num):

prime\_count += 1

else:

composite\_count += 1

return composite\_count, prime\_count

# Sample Input

input\_numbers = input("Enter the numbers separated by spaces: ")

numbers = input\_numbers.split()

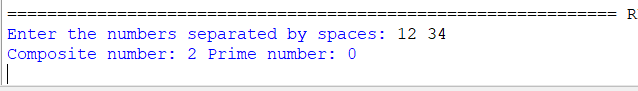
# Count composite and prime numbers

composite\_count, prime\_count = count\_numbers(numbers)

# Sample Output

print(f"Composite number: {composite\_count} Prime number: {prime\_count}")

output:



17

Sample Input:

M = 50

N = 100

K = 7

Sample Output:

50, 57, 64, 71,

Test cases:

M = 15, N = 05, K = 02

.M = 25, N = 50, K = 04

M = 15, N = 100, K = -02

M = 0 , N = 0 , K = 2

M = 200 , N = 200 , K = 50

Code:  
def generate\_sequence(M, N, K):

result = []

# Ensure M is less than or equal to N to generate the sequence

if M > N:

return result

# Generate the sequence

while M <= N:

result.append(M)

M += K

return result

# Sample Input

M = 50

N = 100

K = 7

# Generate sequence

sequence = generate\_sequence(M, N, K)

# Sample Output

print(", ".join(map(str, sequence)))

output:



18)

Write a program for matrix addition?

Sample Input:

Mat1 = 1 2

5 3

Mat2 = 2 3

4 1

Sample Output:

Mat Sum = 3 5

9 4

Code:

def matrix\_addition(mat1, mat2):

rows = len(mat1)

cols = len(mat1[0]) # Assuming both matrices are of the same size

result = [[0]\*cols for \_ in range(rows)]

for i in range(rows):

for j in range(cols):

result[i][j] = mat1[i][j] + mat2[i][j]

return result

# Sample Input

Mat1 = [

[1, 2],

[5, 3]

]

Mat2 = [

[2, 3],

[4, 1]

]

# Perform matrix addition

MatSum = matrix\_addition(Mat1, Mat2)

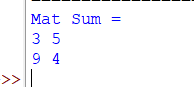
# Sample Output

print("Mat Sum =")

for row in MatSum:

print(" ".join(map(str, row)))

output:



19

Write a program that would sort a list of names in alphabetical order Ascending or Descending, choice get from the user?

Sample Input:

Banana, Carrot, Radish, Apple Jack

Order(A/D) : A

Sample Output:

Apple Banana Carrot Jack Radish

Code:

def sort\_names(names, order):

if order == 'A':

sorted\_names = sorted(names)

elif order == 'D':

sorted\_names = sorted(names, reverse=True)

else:

raise ValueError("Invalid order type. Please choose 'A' for Ascending or 'D' for Descending.")

return sorted\_names

# Sample Input

names = ["Banana", "Carrot", "Radish", "Apple Jack"]

order = input("Order(A/D): ").strip().upper() # Get user input for order

# Sort names

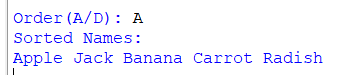
sorted\_names = sort\_names(names, order)

# Sample Output

print("Sorted Names:")

print(" ".join(sorted\_names))

output:



20

Write a program to print the multiplication table of number m up to n.

Sample Input:

M = 4

N = 5

Sample Output:

1x4=4

2x4=8

3x4=12

4x4=16

5x4=20

Test cases:

M = 6, N = -3

M = -3, N = 5

M = 4, N = 0

M = 0, N = 0

M = -5, N = -5

Code:

def multiplication\_table(M, N):

if N < 1:

print("N should be a positive integer. Please provide a valid N.")

return

for i in range(1, N + 1):

result = M \* i

print(f"{i}x{M}={result}")

# Sample Input

M = 4

N = 5

# Print multiplication table

multiplication\_table(M, N)

output:

