DAY–4

1. PERFECT NUMBER

def is\_perfect\_number(n):

if n < 1:

return False

divisors\_sum = 0

for i in range(1, n):

if n % i == 0:

divisors\_sum += i

return divisors\_sum == n

number = int(input("Enter a number: "))

if is\_perfect\_number(number):

print(f"{number} is a perfect number.")

else:

print(f"{number} is not a perfect number.")

2.TRANSPOSAL MATRIX

def print\_matrix(matrix):

for row in matrix:

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

def transpose\_matrix(matrix):

rows = len(matrix)

cols = len(matrix[0])

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

for i in range(rows):

for j in range(cols):

transposed[j][i] = matrix[i][j]

return transposed

matrix = [

[1, 2, 3],

[4, 5, 6],

[7, 8, 9]

]

print("Original Matrix:")

print\_matrix(matrix)

transposed\_matrix = transpose\_matrix(matrix)

print("\nTransposed Matrix:")

print\_matrix(transposed\_matrix)

3.ROW AND COLUMN

def print\_matrix(matrix):

for row in matrix:

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

def row\_sums(matrix):

return [sum(row) for row in matrix]

def column\_sums(matrix):

num\_cols = len(matrix[0])

return [sum(matrix[row][col] for row in range(len(matrix))) for col in range(num\_cols)]

def diagonal\_sums(matrix):

primary\_diagonal = sum(matrix[i][i] for i in range(len(matrix)))

secondary\_diagonal = sum(matrix[i][len(matrix) - i - 1] for i in range(len(matrix)))

return primary\_diagonal, secondary\_diagonal

matrix = [

[1, 2, 3],

[4, 5, 6],

[7, 8, 9]

]

print("Matrix:")

print\_matrix(matrix)

row\_sums\_result = row\_sums(matrix)

column\_sums\_result = column\_sums(matrix)

primary\_diagonal\_sum, secondary\_diagonal\_sum = diagonal\_sums(matrix)

print("\nRow sums:", row\_sums\_result)

print("Column sums:"4. BOUNDARY ELEMENT OF THE MATRIX

def print\_matrix(matrix):

for row in matrix:

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

def boundary\_sum(matrix):

if not matrix or not matrix[0]:

return 0

rows = len(matrix)

cols = len(matrix[0])

total\_sum = 0

for col in range(cols):

total\_sum += matrix[0][col]

if rows > 1:

for col in range(cols):

total\_sum += matrix[rows - 1][col]

for row in range(1, rows - 1):

total\_sum += matrix[row][0] # Left column

if cols > 1:

total\_sum += matrix[row][cols - 1] # Right column

return total\_sum

matrix = [

[1, 2, 3],

[4, 5, 6],

[7, 8, 9]

]

print("Matrix:")

print\_matrix(matrix)

boundary\_sum\_result = boundary\_sum(matrix)

print("\nSum of boundary elements:", boundary\_sum\_result)

, column\_sums\_result)

print("Primary diagonal sum:", primary\_diagonal\_sum)

print("Secondary diagonal sum:", secondary\_diagonal\_sum)

4. BOUNDARY ELEMENT OF THE MATRIX

def print\_matrix(matrix):

for row in matrix:

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

def boundary\_sum(matrix):

if not matrix or not matrix[0]:

return 0

rows = len(matrix)

cols = len(matrix[0])

total\_sum = 0

for col in range(cols):

total\_sum += matrix[0][col]

if rows > 1:

for col in range(cols):

total\_sum += matrix[rows - 1][col]

for row in range(1, rows - 1):

total\_sum += matrix[row][0] # Left column

if cols > 1:

total\_sum += matrix[row][cols - 1] # Right column

return total\_sum

matrix = [

[1, 2, 3],

[4, 5, 6],

[7, 8, 9]

]

print("Matrix:")

print\_matrix(matrix)

boundary\_sum\_result = boundary\_sum(matrix)

print("\nSum of boundary elements:", boundary\_sum\_result)

5. SPIRAL ORDER

def spiral\_order(matrix):

if not matrix:

return []

result = []

top, bottom = 0, len(matrix) - 1

left, right = 0, len(matrix[0]) - 1

while top <= bottom and left <= right:

for i in range(left, right + 1):

result.append(matrix[top][i])

top += 1

for i in range(top, bottom + 1):

result.append(matrix[i][right])

right -= 1

if top <= bottom:

for i in range(right, left - 1, -1):

result.append(matrix[bottom][i])

bottom -= 1

if left <= right:

for i in range(bottom, top - 1, -1):

result.append(matrix[i][left])

left += 1

return result

matrix = [

[1, 2, 3, 4],

[5, 6, 7, 8],

[9, 10, 11, 12],

[13, 14, 15, 16]

]

print("Matrix:")

for row in matrix:

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

spiral\_order\_result = spiral\_order(matrix)

print("\nMatrix in spiral order:")

print(" ".join(map(str, spiral\_order\_result)))

6. SUM OF THE N NUMBERS

def sum\_of\_n\_numbers():

n = int(input("Enter the number of elements: "))

total\_sum = 0

for i in range(n):

num = float(input(f"Enter number {i+1}: "))

total\_sum += num

print(f"The sum of the {n} numbers is: {total\_sum}")

sum\_of\_n\_numbers()

7. SUM OF THR FACTORAL

def factorial(num):

"""Function to calculate the factorial of a number."""

if num == 0 or num == 1:

return 1

else:

return num \* factorial(num - 1)

def sum\_of\_factorials(n):

"""Function to calculate the sum of factorials from 1! to n!."""

total\_sum = 0

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

total\_sum += factorial(i)

return total\_sum

n = int(input("Enter a number: "))

result = sum\_of\_factorials(n)

print(f"The sum of factorials from 1! to {n}! is: {result}")

8. SUM OF THE SQUARE ROOTS

def sum\_of\_squares(n):

"""Function to calculate the sum of squares from 1^2 to n^2."""

total\_sum = sum(i\*\*2 for i in range(1, n + 1))

return total\_sum

n = int(input("Enter a number: "))

result = sum\_of\_squares(n)

print(f"The sum of squares from 1^2 to {n}^2 is: {result}")

**9. MEAN MEDIUM AND MODE OF THE ELEMNT**

from statistics import mean, median, mode, StatisticsError

def calculate\_statistics(numbers):

"""Function to calculate mean, median, and mode of a list of numbers."""

try:

mean\_value = mean(numbers)

except StatisticsError:

mean\_value = None

try:

median\_value = median(numbers)

except StatisticsError:

median\_value = None

try:

mode\_value = mode(numbers)

except StatisticsError:

mode\_value = None

return mean\_value, median\_value, mode\_value

numbers = [1, 2, 2, 3, 4, 4, 4, 5, 6]

mean\_value, median\_value, mode\_value = calculate\_statistics(numbers)

print(f"Mean: {mean\_value}")

print(f"Median: {median\_value}")

print(f"Mode: {mode\_value}")

**10. nth LARGEST NUMBER**

def nth\_largest(numbers, n):

"""Function to find the nth largest number in a list."""

if n > len(numbers):

return None

sorted\_numbers = sorted(numbers, reverse=True)

return sorted\_numbers[n - 1]

numbers = [4, 2, 5, 1, 3, 7, 6]

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

result = nth\_largest(numbers, n)

if result is not None:

print(f"The {n}th largest number is: {result}")

else:

print(f"The list does not have {n} elements.")