Day 1 lab programs

1.

def isomorphic\_strings(s, t):

if len(s) != len(t):

return False

mapping = {}

used\_chars = set()

for i in range(len(s)):

if s[i] in mapping:

if mapping[s[i]] != t[i]:

return False

else:

if t[i] in used\_chars:

return False

mapping[s[i]] = t[i]

used\_chars.add(t[i])

return True

# Test cases

print(isomorphic\_strings("egg", "add")) # Output: True

print(isomorphic\_strings("foo", "bar")) # Output: False

print(isomorphic\_strings("paper", "title")) # Output: True

print(isomorphic\_strings("fry", "sky")) # Output: True

print(isomorphic\_strings("apples", "apple")) # Output: False

2.

def sumsquare(l):

odd\_sum = 0

even\_sum = 0

for num in l:

if num % 2 == 0:

even\_sum += num \*\* 2

else:

odd\_sum += num \*\* 2

return [odd\_sum, even\_sum]

# Sample Input

num\_elements = int(input("Enter the number of elements: "))

elements = []

for \_ in range(num\_elements):

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

elements.append(element)

# Calculate and print the output

output = sumsquare(elements)

print(output)

3.

function isHappy(n):

seen = set() # Keep track of seen numbers to detect cycles

while n != 1 and n not in seen:

seen.add(n)

n = calculate\_next(n)

return n == 1

function calculate\_next(n):

next\_num = 0

while n > 0:

digit = n % 10

next\_num += digit \* digit

n //= 10

return next\_num

print(isHappy(19)) # Output: True

print(isHappy(2)) # Output: False

print(isHappy(-1)) # Output: False

print(isHappy(0)) # Output: False

print(isHappy(5)) # Output: False

4.

function isPalindrome(x):

if x < 0:

return False

original\_x = x

reversed\_x = 0

while x > 0:

digit = x % 10

reversed\_x = reversed\_x \* 10 + digit

x //= 10

return original\_x == reversed\_x

print(isPalindrome(121)) # Output: True

print(isPalindrome(-121)) # Output: False

print(isPalindrome(10)) # Output: False

print(isPalindrome("abc")) # Output: False (Invalid input)

print(isPalindrome(0)) # Output: True

5.

def calculate\_price(fresh\_loaves, day\_old\_loaves):

price\_per\_loaf = 185

discount\_percentage = 60

fresh\_loaves\_price = fresh\_loaves \* price\_per\_loaf

day\_old\_loaves\_price = day\_old\_loaves \* price\_per\_loaf \* (1 - discount\_percentage / 100)

total\_price = fresh\_loaves\_price + day\_old\_loaves\_price

return fresh\_loaves\_price, day\_old\_loaves\_price, total\_price

# Input

fresh\_loaves = int(input("Enter the number of fresh loaves purchased: "))

day\_old\_loaves = int(input("Enter the number of day old loaves purchased: "))

# Calculate prices

fresh\_price, day\_old\_price, total\_price = calculate\_price(fresh\_loaves, day\_old\_loaves)

# Output

print("Regular price: Rs.{:.2f}".format(fresh\_price))

print("Amount of new loaves: Rs.{:.2f}".format(fresh\_price))

print("Amount of day old loaves: Rs.{:.2f}".format(day\_old\_price))

print("Total amount: Rs.{:.2f}".format(total\_price))

print("Test Case 1:")

calculate\_price(4, 6) # Output will be calculated and displayed

print("Test Case 2:")

calculate\_price(-1, 5) # Output will be calculated and displayed

print("Test Case 3:")

calculate\_price(0, 6) # Output will be calculated and displayed

print("Test Case 4:")

calculate\_price(7, 8) # Output will be calculated and displayed

print("Test Case 5:")

calculate\_price(3, 4) # Output will be calculated and displayed

6.

def max\_area(height):

left = 0

right = len(height) - 1

max\_area = 0

while left < right:

width = right - left

h = min(height[left], height[right])

area = width \* h

max\_area = max(max\_area, area)

if height[left] < height[right]:

left += 1

else:

right -= 1

return max\_area

# Test cases

print(max\_area([1, 5, 4, 3])) # Output: 6

print(max\_area([3, 1, 2, 4, 5])) # Output: 12

print(max\_area([1, 8, 6, 2, 5, 4, 8, 3, 7])) # Output: 49

print(max\_area([1, 1])) # Output: 1

print(max\_area([7, 3])) # Output: 3

7.

def count\_sorted\_vowel\_strings(n):

# Using dynamic programming approach

dp = [[0] \* 5 for \_ in range(n+1)]

for i in range(5):

dp[1][i] = 1

for i in range(2, n+1):

for j in range(5):

dp[i][j] = sum(dp[i-1][k] for k in range(j, 5))

return sum(dp[n])

# Test cases

print(count\_sorted\_vowel\_strings(1)) # Output: 5

print(count\_sorted\_vowel\_strings(2)) # Output: 15

print(count\_sorted\_vowel\_strings(33)) # Output: 66045

print(count\_sorted\_vowel\_strings(-5)) # Output: 0 (Invalid input)

print(count\_sorted\_vowel\_strings(10)) # Output: 1716

8.

def isNumber(s):

try:

float(s) # Try converting the string to a float

return True

except ValueError:

return False

# Test cases

print(isNumber("0")) # Output: True

print(isNumber("e")) # Output: False

print(isNumber(" ")) # Output: False

print(isNumber(".")) # Output: False

print(isNumber("%")) # Output: False

9.

def max\_guests\_on\_cruise(T, E, L):

events = []

for i in range(len(E)):

events.append((E[i], 1)) # Entering guest

events.append((L[i], -1)) # Leaving guest

events.sort() # Sort events based on time

max\_guests = 0

current\_guests = 0

for event in events:

current\_guests += event[1]

max\_guests = max(max\_guests, current\_guests)

return max\_guests

# Sample Input

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

E = []

L = []

for \_ in range(T):

enter = int(input())

E.append(enter)

for \_ in range(T):

leave = int(input())

L.append(leave)

# Calculate and print the output

output = max\_guests\_on\_cruise(T, E, L)

print(output)

print(max\_guests\_on\_cruise(-4, [1, 5, 9, 10], [0, 2, 3, 4])) # Output: 0 (Invalid input)

print(max\_guests\_on\_cruise(0, [10, 2, 3, 4], [1234])) # Output: 0 (Invalid input)

print(max\_guests\_on\_cruise(4, [12, 85], [100])) # Output: 2

print(max\_guests\_on\_cruise(5, [42, 0, 35, 12, 15], [1, 2, 1, 3, 4])) # Output: 6

print(max\_guests\_on\_cruise(1, [12], [10])) # Output: 2

10.

def modify\_string(s):

freq = [0] \* 26 # Initialize frequency array for alphabets

for char in s:

freq[ord(char) - ord('a')] += 1

result = []

for char in s:

new\_char = chr(((ord(char) - ord('a') + freq[ord(char) - ord('a')]) % 26) + ord('a'))

result.append(new\_char)

return ''.join(result)

# Test cases

print(modify\_string("ghee")) # Output: hggi

print(modify\_string("elephant")) # Output: epehdsgf

print(modify\_string("apple")) # Output: anppe

print(modify\_string("orange")) # Output: opsbwh

print(modify\_string("lion")) # Output: lzqn