Day 5 lab python programs

2.def calculate\_bonus(salary, grade):

bonus\_percentage = 0

if grade == 'A':

bonus\_percentage = 0.05

elif grade == 'B':

bonus\_percentage = 0.10

if salary < 10000:

bonus\_percentage += 0.02

bonus = salary \* bonus\_percentage

total\_salary = salary + bonus

return bonus, total\_salary

# Test cases

for \_ in range(5):

grade = input("Enter the grade of the employee: ")

salary = float(input("Enter the employee salary: "))

bonus, total\_salary = calculate\_bonus(salary, grade)

print(f"Salary={salary}")

print(f"Bonus={bonus}")

print(f"Total to be paid: {total\_salary}\n")

1.

def length\_of\_last\_word(s):

words = s.strip().split() # Remove leading/trailing spaces and split the string

if words:

return len(words[-1])

return 0

# Test cases

test\_strings = [

"Hello World",

" fly me to the moon ",

"luffy is still joyboy",

"123",

" 45&29 8\*6^4"

]

for s in test\_strings:

result = length\_of\_last\_word(s)

print(f"Input: {s}")

print(f"Output: {result}\n")

3.

def numSquares(n):

dp = [float('inf')] \* (n + 1)

dp[0] = 0

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

j = 1

while j \* j <= i:

dp[i] = min(dp[i], dp[i - j \* j] + 1)

j += 1

return dp[n]

# Test cases

test\_cases = [12, 13, 1, 4, 3]

for n in test\_cases:

result = numSquares(n)

print(f"Input: n = {n}")

print(f"Output: {result}\n")

4.

def check\_divisibility(num1, num2):

prod = num1 \* num2

sum\_nums = num1 + num2

if prod % sum\_nums == 0:

return "YEAH"

else:

return "NAH"

# Test cases

test\_cases = [

(60, 48),

(4, 8),

(-10, 0),

(12, 34),

(16, 17)

]

for num1, num2 in test\_cases:

result = check\_divisibility(num1, num2)

print(f"Enter the numbers: {num1} {num2}")

print(f"Output: {result}\n")

5.def find\_peak\_element(nums):

left, right = 0, len(nums) - 1

while left < right:

mid = left + (right - left) // 2

if nums[mid] < nums[mid + 1]:

left = mid + 1

else:

right = mid

return left

# Test cases

test\_cases = [

[1, 2, 3, 1],

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

[5, 10, 20, 15],

[1, 3, 2, 4, 1, 0],

[1, 1, 1, 3, 2, 1, 2]

]

for nums in test\_cases:

result = find\_peak\_element(nums)

print(f"Input: {nums}\nOutput: {result}\n")

7.

def longest\_substring(s, k):

if not s:

return 0

# Count the frequency of each character in the string

char\_count = {}

for char in s:

char\_count[char] = char\_count.get(char, 0) + 1

# Find the index of the first character that appears less than k times

for i, char in enumerate(s):

if char\_count[char] < k:

left\_part = longest\_substring(s[:i], k)

right\_part = longest\_substring(s[i + 1:], k)

return max(left\_part, right\_part)

# All characters appear at least k times

return len(s)

# Test cases

test\_cases = [

("aaabb", 3),

("ababbc", 2),

("ababab", 3),

("xyzxyz", 2),

("12345", 0)

]

for s, k in test\_cases:

result = longest\_substring(s, k)

print(f"Input: s = '{s}', k = {k}\nOutput: {result}\n")

8.

def min\_swaps\_to\_chessboard(board):

n = len(board)

# Count the number of 1s and 0s in the first row and first column

row\_count = [0] \* n

col\_count = [0] \* n

for i in range(n):

for j in range(n):

if board[i][j] != (i + j) % 2:

return -1

row\_count[i] += board[i][j]

col\_count[j] += board[i][j]

# Check if it's possible to form a valid chessboard

for count in (row\_count, col\_count):

if sorted(count) != [n // 2, (n + 1) // 2]:

return -1

# Calculate the number of swaps needed for rows and columns

row\_diff = sum([abs(row\_count[i] - n // 2) for i in range(n)]) // 2

col\_diff = sum([abs(col\_count[i] - n // 2) for i in range(n)]) // 2

return (row\_diff + col\_diff) // 2

# Test cases

test\_cases = [

[[0,1,1,0],[0,1,1,0],[1,0,0,1],[1,0,0,1]],

[[0,1],[1,0]],

[[1,0],[1,0]],

[[0, 1, 0], [1, 0, 1], [1, 1, 0]],

[[0, 1, 1, 0], [0, 1, 1, 0], [1, 0, 0, 1], [1, 0, 0, 1]]

]

for board in test\_cases:

result = min\_swaps\_to\_chessboard(board)

print(f"Input: board = {board}\nOutput: {result}\n")

9.

def shuffle(l1, l2):

result = []

min\_length = min(len(l1), len(l2))

for i in range(min\_length):

result.append(l1[i])

result.append(l2[i])

result.extend(l1[min\_length:])

result.extend(l2[min\_length:])

return result

# Test cases

test\_cases = [

([0, 4], [-1, 1]),

([5], [5]),

([10], [5]),

([6], [9])

]

for l1, l2 in test\_cases:

shuffled\_list = shuffle(l1, l2)

print(f"Shuffled list = {shuffled\_list}")

Shuffled list = [0, -1, 4, 1]

Shuffled list = [5]

Shuffled list = [10, 5]

Shuffled list = [6, 9]

10.

def reverse\_words(s):

words = s.split()

reversed\_words = ' '.join(reversed(words))

return reversed\_words

# Test cases

test\_cases = [

"the sky is blue",

" hello world ",

"a good example",

"apple is red",

"Red rose"

]

for s in test\_cases:

result = reverse\_words(s)

print(f"Input: '{s}'\nOutput: '{result}'\n")

Input: 'the sky is blue'

Output: 'blue is sky the'

Input: ' hello world '

Output: 'world hello'

Input: 'a good example'

Output: 'example good a'

Input: 'apple is red'

Output: 'red is apple'

Input: 'Red rose'

Output: 'rose Red'

6.

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

for i in range(n):

print(' '\*(n-i), end='')

print(' '.join(map(str, str(11\*\*i))))