**CSA0888 – PYTHON**

**PROGRAMMING**

**ASSIGNMENT – 5**

**1.LENGTH OF LAST WORD**

**INPUT:**

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

words = s.split() if not words:

return 0

last\_word = words[-1] return len(last\_word)

input\_string = "my name is " result = length\_of\_last\_word(input\_string) print(result)

**2.EMPLOYEE INPUT:**

def calculate\_salary\_with\_bonus(salary, grade):

bonus\_percentage = 0 if salary < 10000: bonus\_percentage += 0.02

if grade == 'A': bonus\_percentage += 0.05

elif grade == 'B':

bonus\_percentage += 0.1

bonus = salary \* bonus\_percentage final\_salary = salary + bonus return final\_salary

salary = float(input("Enter the salary: ")) grade = input("Enter the grade (A or B): ") final\_salary = calculate\_salary\_with\_bonus(salary, grade) print(f"The final salary of the employee is ${final\_salary:.2f}")

3. Input:

def numSquares(n):

if n <= 0:

return 0

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] # Example usage

n = 12

print(numSquares(n)) # Output should be 2 (4 + 4)

4.input:

# Number of test cases t = int(input("Enter the number of test cases: ")) for \_ in range(t):

prod, sum = map(int, input("Enter prod and sum: ").split()

# Check if prod is divisible by sum if prod % sum == 0: print("YEAH") else:

print("NAH")

5.input:

def find\_peak\_element(nums): left, right = 0, len(nums) while left < right:

mid = left + (right - left) // 2 if nums[mid] > nums[mid + 1]:

right = mid else:

left = mid + 1

return left

# Example usage nums = [1, 2, 3, 1] peak\_index = find\_peak\_element(nums) print("Peak element is at index:",

6.input:

def binomial\_coefficient(n, k): if k == 0 or k == n: return 1

return binomial\_coefficient(n - 1, k - 1) + binomial\_coefficient(n - 1, k)

def generate\_binomial\_triangle(rows):

triangle = [] for n in range(rows): row = [binomial\_coefficient(n, k) for k in range(n + 1)] triangle.append(row)

return triangle

def sum\_of\_row(triangle, n):

return sum(triangle[n])

n = 5

triangle = generate\_binomial\_triangle(n + 1) row\_sum = sum\_of\_row(triangle, n) print("Binomial Triangle:") for row in triangle:

print(row)

print(f"Sum of elements in the {n}th row: {row\_sum}")

7.input:

def longestSubstring(s, k):

if len(s) == 0 or k > len(s):

return 0

char\_count = {} for char in s:

if char in char\_count:

char\_count[char] += 1 else:

char\_count[char] = 1

for char in char\_count:

if char\_count[char] < k: splitting\_char = char break

else: return len(s)

substrings = s.split(splitting\_char) return max(longestSubstring(substring, k) for substring in substrings)

# Example usage

s = "aaabb" k = 3

print(longestSubstring(s, k)) # Output: 3

8.input:

def min\_swaps\_to\_chessboard(board): n = len(board) ones = sum(sum(row) for row in board) zeros = n \* n - ones if abs(ones - zeros) > 1: return -1

row\_swap\_candidates = 0 col\_swap\_candidates = 0 for i in range(n): row = board[i] col = [board[j][i] for j in range(n)]

if row.count(1) != n - row.count(1): return -1

if row[0] == 1: row\_swap\_candidates += 1

if col[0] == 1:

col\_swap\_candidates += 1

min\_swaps = float('inf') if n % 2 == 0:

min\_swaps = min(min\_swaps, row\_swap\_candidates // 2) min\_swaps = min(min\_swaps, col\_swap\_candidates // 2) else:

min\_swaps = min(min\_swaps, row\_swap\_candidates) min\_swaps = min(min\_swaps, col\_swap\_candidates)

return min\_swaps

# Example input matrix = [

[0, 1, 0],

[1, 0, 1],

[0, 1, 0]

]

result = min\_swaps\_to\_chessboard(matrix) print(result) # Output: 0 or 1 (depending on the valid chessboard configuration)

9.input:

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

# Example usage list1 = [1, 3, 5, 7] list2 = [2, 4, 6, 8, 10] shuffled\_list = shuffle(list1, list2) print(shuffled\_list) # Output: [1, 2, 3, 4, 5, 6, 7, 8, 10]

10.input:

def reverse\_words(s): words = s.split() reversed\_string = ' '.join(reversed(words)) return reversed\_string