### Programming Guide for “Arrays and Linked Lists”

#### 1. Array Sum

**Overview:** Calculate the sum of elements in an array.

**Pseudo Code:**

FUNCTION array\_sum(arr)  
 INITIALIZE sum to 0  
 FOR each element in arr  
 ADD element to sum  
 RETURN sum  
END FUNCTION

**Implementation Guide:** - Use a loop to iterate through each element of the array. - Accumulate the sum of these elements. - Return the final sum.

#### 2. Find Middle Node in a Linked List

**Overview:** Find the middle node of a singly linked list.

**Pseudo Code:**

FUNCTION find\_middle\_node(linked\_list)  
 INITIALIZE slow and fast pointers to the head of linked\_list  
 WHILE fast and fast.next are not None  
 MOVE slow one step forward  
 MOVE fast two steps forward  
 RETURN slow (middle node)  
END FUNCTION

**Implementation Guide:** - Use two pointers: slow and fast. slow moves one node at a time, while fast moves two nodes. - When fast reaches the end of the list, slow will be at the middle. - Return the node pointed to by slow.

#### 3. Remove Duplicates from Sorted Array

**Overview:** Remove duplicates from a sorted array in-place.

**Pseudo Code:**

FUNCTION remove\_duplicates\_from\_sorted\_array(arr)  
 IF arr is empty  
 RETURN 0  
 INITIALIZE write\_index to 1  
 FOR i from 1 to length of arr  
 IF arr[i] is not equal to arr[i - 1]  
 SET arr  
  
[write\_index] to arr[i]  
 INCREMENT write\_index  
 RETURN write\_index  
END FUNCTION

**Implementation Guide:** - Check if the array is empty. If it is, return 0. - Use a write index to track the position where the next unique element should be written. - Iterate through the array and when you find a unique element (not equal to its predecessor), copy it to the write index and increment the write index. - Return the new length of the array (write index).