# **Analysis:**

## **Analyze the time complexity of each operation.**

The time complexity analysis for each operation in the singly linked list implementation of the task management system:

1. Add Task

* Operation: Adding a task to the end of the list.
* Time Complexity: O(n)  
  (In the worst case, we need to traverse the entire list to reach the end.)

2. Search Task

* Operation: Searching for a task by taskId.
* Time Complexity: O(n)  
  (In the worst case, we may need to traverse the entire list to find the task or determine it’s not present.)

3. Traverse Tasks

* Operation: Traversing all tasks to display them.
* Time Complexity: O(n)  
  (We need to visit each node in the list.)

4. Delete Task

* Operation: Deleting a task by taskId.
* Time Complexity: O(n)  
  (In the worst case, we may need to traverse the entire list to find the task to delete.)

## **Discuss the advantages of linked lists over arrays for dynamic data.**

The key advantages of linked lists over arrays for dynamic data:

1. Dynamic Size

* Linked Lists: Can grow or shrink in size dynamically, allowing efficient memory usage.
* Arrays: Have a fixed size, requiring resizing or reallocating when capacity is exceeded.

2. Efficient Insertions/Deletions

* Linked Lists: Inserting or deleting nodes can be done in O(1) time if the position is known (no need to shift elements).
* Arrays: Insertions and deletions require O(n) time as elements must be shifted.

3. Memory Utilization

* Linked Lists: Allocate memory for each node as needed, leading to more efficient memory use when the number of elements fluctuates.
* Arrays: May lead to wasted memory space if the allocated size is larger than needed.

4. No Contiguous Memory Requirement

* Linked Lists: Nodes can be stored anywhere in memory, making them less prone to fragmentation.
* Arrays: Require contiguous memory blocks, which can be a limitation in fragmented memory environments.