Exercise 5: Task Management System

Analysis:

- Analyze the time complexity of each operation.
- Discuss the advantages of linked lists over arrays for dynamic data.

Time Complexity:

Operation	**Time complexity**	**Reason**
Add(at head)	O(1)	Just update pointers to add a node at the beginning.
Add(at tail)	O(n)	Requires traversal to the end unless tail is tracked.
Search	O(n)	Linear search needed to find a task by name, ID, or status.
Traverse	O(n)	Visit each task node sequentially.
Delete	O(n)	Need to find the node and update the previous node's pointer.

Advantages of Linked Lists over Arrays for Dynamic Data

Advantage	**Explanation**
Dynamic size	No need to declare a fixed size. You can grow/shrink the list as tasks are added/removed.
Efficient insertion/deletion	Especially at the beginning or middle—no need to shift elements like in arrays.
No memory reallocation	Unlike arrays, you don't need to allocate a larger block of memory when expanding.
Better suited for frequently changing data	Great for task queues or schedulers where tasks are added/removed often.

Conclusion

- Use a **linked list** when the number of tasks is dynamic and frequent insertions/deletions are needed.
- Use **arrays** only if task count is fixed and random access (by index) is critical.
- For advanced systems (e.g., with priority), you might consider **priority queues**, **doubly linked lists**, or even **heaps**.