**Exercise 5: Task Management System**

Scenario:

You are developing a task management system where tasks need to be added, deleted, and traversed efficiently.

* Explain the different types of linked lists (Singly Linked List, Doubly Linked List).

->

Singly Linked List:

A singly linked list is a type of linked list where each node contains two components: the data and a reference to the next node in the sequence. This means that each node points only to the next node, creating a unidirectional link from the start of the list (the head) to the end (where the last node points to null).

In a singly linked list, we can start at the head node and follow the next references to traverse the entire list. However, since each node only knows about the next node, you cannot easily traverse backward or access the previous node directly. This can make certain operations, such as deletion, more complex because you need to find the previous node to adjust its reference.

Doubly Linked List:

A doubly linked list extends the concept of a singly linked list by adding an additional reference to each node. Each node in a doubly linked list contains three components: the data, a reference to the next node, and a reference to the previous node. This setup allows traversal in both directions—forward and backward.

With a doubly linked list, you can navigate from the head node to the end of the list by following the next references, or you can traverse backwards from the tail node to the head by following the previous references. This bidirectional capability simplifies operations such as deletion, as you can directly access both the next and previous nodes, making it easier to update references when removing nodes.

Analysis:

o Analyze the time complexity of each operation.

**Time Complexity of Operations in a Linked List**

1. **Add Operation**
   * **At the Beginning:** O(1) - Adding a node at the start of the list involves updating the head pointer and linking the new node.
   * **At the End:** O(n) - To add a node at the end, you need to traverse the entire list to find the last node, which requires O(n) time. However, if you maintain a tail pointer, this operation can be O(1).
   * **At a Specific Position:** O(n) - You need to traverse the list to find the insertion point, which takes O(n) time.
2. **Traverse Operation**
   * **General Traversal:** O(n)- To traverse the entire list from head to tail, you need to visit each node once.
3. **Delete Operation**
   * **From the Beginning:** O(1) - Removing the first node involves updating the head pointer to point to the next node.
   * **From the End:** O(n) - Deleting the last node requires traversal to find the second-to-last node so that it can update its next pointer to null. If you have a tail pointer, but not a backward link, this operation is still O(n)
   * **From a Specific Position:** O(n) - Similar to adding at a specific position, finding the node to delete requires traversal, so the operation is O(n).

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

* The advantages of linked lists over arrays in short:

1. **Dynamic Size:** Linked lists can easily grow or shrink without needing to resize.
2. **Efficient Insertions/Deletions:** Adding or removing elements is quicker, especially in the middle, without shifting elements.
3. **Memory Usage:** Can be more memory-efficient for dynamic data, though they have overhead for storing pointers.
4. **No Contiguous Memory Requirement:** Nodes can be scattered in memory, avoiding the need for large contiguous memory blocks.