**Exercise 5: Task Management System**

**Scenario:**

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

**Steps:**

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

A singly linked list is a data structure consisting of a sequence of nodes where each node contains data and a reference (or link) to the next node in the sequence. The last node's reference points to null, indicating the end of the list.

Advantages:

* Simple to implement.
* Efficient insertion and deletion operations, especially at the beginning of the list.

Disadvantages:

* No efficient way to access elements directly (no random access).
* Extra memory space is required for storing the reference to the next node.

Doubly Linked List

A doubly linked list is a data structure where each node contains data, a reference to the next node, and a reference to the previous node. This allows traversal in both directions (forward and backward).

Advantages:

* Allows efficient bidirectional traversal.
* Simplifies deletion operations since the previous node can be accessed directly.

Disadvantages:

* More complex to implement than a singly linked list.
* Requires more memory for storing an additional reference to the previous node.

1. **Setup:**
   * Create a class **Task** with attributes like **taskId**, **taskName**, and **status**.
2. **Implementation:**
   * Implement a singly linked list to manage tasks.
   * Implement methods to **add**, **search**, **traverse**, and **delete** tasks in the linked list.
3. **Analysis:**
   * Analyze the time complexity of each operation.

For the singly linked list implementation provided:

* **Add a Task:**
  + **Time Complexity:** O(n) (since we need to traverse to the end of the list to insert the new task)
* **Search for a Task:**
  + **Time Complexity:** O(n) (since we need to potentially traverse the entire list to find the task)
* **Traverse Tasks:**
  + **Time Complexity:** O(n) (since we need to visit each node in the list)
* **Delete a Task:**
  + **Time Complexity:** O(n) (since we need to potentially traverse the entire list to find the task to delete)

Discuss the advantages of linked lists over arrays for dynamic data

* **Dynamic Size:** Linked lists can grow and shrink dynamically, allowing efficient memory use without needing to reallocate or resize the entire structure as with arrays.
* **Efficient Insertions/Deletions:** Inserting or deleting elements at the beginning or middle of a linked list is more efficient compared to arrays, which require shifting elements.