**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).

***Singly Linked List:***

*Structure:* Consists of nodes where each node contains a data part and a reference (or pointer) to the next node in the list.

*Operations:* Efficient insertions and deletions at the beginning. Traversal is required for operations at the end or middle.

*Use Case:* Useful for implementing stacks, queues, and other linear data structures.

***Doubly Linked List:***

*Structure:* Each node contains a data part, a reference to the next node, and a reference to the previous node.

*Operations:* Allows bidirectional traversal which make insertions and deletions more efficient.

*Use Case:* Useful for implementing complex data structures like deques, and for applications where bidirectional traversal is needed.

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.

1. **Add Task:** O(n) for adding a task at the end since it requires traversal to the end of the list.
2. **Search Task:** O(n) as it requires traversal through the list to find the task.
3. **Traverse Tasks:** O(n) as it involves visiting each node in the list.
4. **Delete Task:** O(n) as it may require traversal to find the task to be deleted.

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

1. Dynamic Size: Linked lists can grow and shrink in size dynamically, unlike arrays which have a fixed size.
2. Efficient Insertions/Deletions: Inserting or deleting elements in the middle of a linked list is more efficient (O(1) time for insertions/deletions at the head) compared to arrays (O(n) time).