**EXERCISE 5**

**Task Management System**

**1. Understand Linked Lists:**

**Types of Linked Lists:**

1. **Singly Linked List:**
   * **Structure**: Consists of nodes where each node contains data and a reference (or pointer) to the next node in the sequence.
   * **Operations:** Can efficiently insert and delete nodes from the beginning or middle, but requires traversal from the start to access or modify nodes elsewhere.
2. **Doubly Linked List:**
   * **Structure:** Similar to a singly linked list, but each node contains two references—one to the next node and one to the previous node.
   * **Operations:** Allows for efficient insertion and deletion from both ends and can traverse in both directions.

**2. Setup:**

Create a class Task with attributes such as taskId, taskName, and status.

**3. Implementation:**

Singly Linked List is a fundamental data structure in computer science and programming, it consists of nodes where each node contains a data field and a reference to the next node in the node. The last node points to null, indicating the end of the list. This linear structure supports efficient insertion and deletion operations, making it widely used in various applications.

**4. Analysis:**

**Time Complexity of Operations:**

* **Add Task:** O(n) - Inserting at the end requires traversal to the end of the list. If the task needs to be added at the beginning, it’s O(1).
* **Search Task:** O(n) - Requires traversal of the list to find the task.
* **Traverse Tasks:** O(n) - Requires visiting each node in the list.
* **Delete Task:** O(n) - Requires traversal to find the task to delete. If the task is at the beginning, it’s O(1).

**Advantages of Linked Lists Over Arrays for Dynamic Data:**

* **Dynamic Size:** Linked lists can dynamically adjust their size without needing to resize or shift elements.
* **Efficient Insertions/Deletions:** Easier to insert and delete nodes without shifting other elements, especially when the position is known.

Linked lists are particularly useful in scenarios where dynamic resizing and frequent insertions and deletions are required, whereas arrays are more suitable when the size is fixed and direct indexing is needed.