**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).

**Types of Linked Lists**

**Singly Linked List**:

* **Structure**: Each node has data and a reference to the next node.
* **Traversal**: Forward only.
* **Memory Usage**: Less memory usage (one reference per node).
* **Operations**: Simple insertion and deletion at the start.

**Doubly Linked List**:

* **Structure**: Each node has data, a reference to the next node, and a reference to the previous node.
* **Traversal**: Forward and backward.
* **Memory Usage**: More memory usage (two references per node).
* **Operations**: Easier insertion and deletion at both ends and in the middle.

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

**Time Complexity**:

* **Add Task**: O(n) (find the end of the list).
* **Search Task**: O(n) (traverse from head to tail).
* **Traverse Tasks**: O(n) (visit each node).
* **Delete Task**: O(n) (search and update references).

**Advantages of Linked Lists Over Arrays**:

* **Dynamic Size**: Grows and shrinks without large memory reallocation.
* **Efficient Insertions/Deletions**: Better for operations at the beginning or middle, without shifting elements.
* **Memory Utilization**: Non-contiguous memory allocation avoids fragmentation.
* **Flexibility**: Easier to implement complex data structures like stacks, queues, and graphs.