**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** : Each node contains data and a link to the next node. Traversal is unidirectional.

**Doubly Linked List:** Each node contains data, a link to the next node, and a link to the previous node.

Traversal can be done in both directions.

1. **Analysis:**
   * **Analyze the time complexity of each operation.**

**Access**: Both singly linked lists and doubly linked lists have a time complexity of **O(n).**

**Search**: Both singly linked lists and doubly linked lists have a time complexity of **O(n).**

**Insert at Beginning:** Both singly linked lists and doubly linked lists have a time complexity of **O(1).**

**Insert at End**: Singly linked lists have a time complexity of O(n), while doubly linked lists have a time complexity of **O(1)**.

**Insert at Middle**: Both singly linked lists and doubly linked lists have a time complexity of **O(n)**.

**Delete at Beginning**: Both singly linked lists and doubly linked lists have a time complexity of **O(1)**.

**Delete at End**: Singly linked lists have a time complexity of O(n), while doubly linked lists have a time complexity of **O(n)**.

**Delete at Middle:** Both singly linked lists and doubly linked lists have a time complexity of **O(n)**.

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

**Advantages of Linked Lists over Arrays for Dynamic Data :** Dynamic Size: Linked lists can grow or shrink dynamically as needed, unlike arrays which have a fixed size.

**Efficient Insertions and Deletions:** Inserting or deleting elements at any position in a linked list is generally more efficient than in an array, especially for large datasets.

**Memory Efficient:** Linked lists only allocate memory for nodes as needed, avoiding potential memory wastage like in arrays.