**ANSWER**

**Exercise 5: Task Management System**

**Scenario:**

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

**Steps:**

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

**Singly Linked List:**

* **Description:** A linear collection of nodes where each node contains a data element and a reference (or link) to the next node in the sequence. The first node is called the head, and the last node points to null.
* **Characteristics:**
  + Simple to implement.
  + Efficient insertion and deletion from the beginning.
* **Use Cases:** Suitable for applications where the list is frequently modified, such as in queue implementations.

**Doubly Linked List:**

* **Description:** A linear collection of nodes where each node contains a data element, a reference to the next node, and a reference to the previous node. This allows traversal in both directions.
* **Characteristics:**
  + Allows for efficient insertion and deletion from both ends.
  + More complex to implement due to additional pointers.
* **Use Cases:** Suitable for applications requiring bidirectional traversal, such as in navigation systems and certain data structures like deques.
* **Analysis:**
* **Analyze the time complexity of each operation.**
  + **Time Complexity of Operations**

1. **Add Task:**
   * Operation: Adding a node to the end of the linked list.
   * Time Complexity: O(n) (traversal to the end of the list).
2. **Search Task:**
   * Operation: Linear search through the linked list.
   * Time Complexity: O(n).
3. **Traverse Tasks:**
   * Operation: Iterating through all nodes of the linked list.
   * Time Complexity: O(n).
4. **Delete Task:**
   * Operation: Linear search followed by pointer adjustment.
   * Time Complexity: O(n).
   * **Discuss the advantages of linked lists over arrays for dynamic data.**

**Dynamic Size**: Linked lists can grow and shrink dynamically, making them more flexible for applications where the number of elements can change.

**Efficient Insertions/Deletions**: Insertions and deletions can be done in constant time (O(1)) if the position is known, without the need to shift elements as in arrays.

**Memory Utilization**: Linked lists use memory efficiently for dynamic data, avoiding the need for resizing arrays.