**Exercise 5: Task Management System**

**Types of Linked Lists:**

* **Singly Linked List:**
  + **Structure:** Each node contains a data part and a reference (or pointer) to the next node in the sequence.
  + **Advantages:** Simple to implement and efficient for sequential access and insertion/deletion at the beginning of the list.
  + **Disadvantages:** Inefficient for backward traversal since it doesn't store references to previous nodes.
* **Doubly Linked List:**
  + **Structure:** Each node contains a data part and two references, one pointing to the next node and the other pointing to the previous node.
  + **Advantages:** Allows efficient forward and backward traversal and easier deletion of nodes.
  + **Disadvantages:** Requires more memory due to the extra reference for the previous node and slightly more complex implementation.

**Time Complexity Analysis:**

* **Add Operation:**
  + **Time Complexity:** O(n) in the worst case. When adding at the end of the list.
  + **Explanation:** Traversal to the end of the list is required, making it linear in complexity.
* **Search Operation:**
  + **Time Complexity:** O(n)
  + **Explanation:** In the worst case, each node might need to be checked, making it linear in complexity.
* **Traverse Operation:**
  + **Time Complexity:** O(n)
  + **Explanation:** Each node needs to be visited once, making it linear in complexity.
* **Delete Operation:**
  + **Time Complexity:** O(n)
  + **Explanation:** In the worst case, traversal to the node to be deleted is required, making it linear in complexity.

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

* **Dynamic Size:** Linked lists can grow and shrink dynamically, making them suitable for applications where the number of elements is not known in advance.
* **Efficient Insertions/Deletions:** Insertions and deletions can be more efficient, especially at the beginning of the list, as they do not require shifting elements as in arrays.
* **Memory Usage:** Linked lists do not require a contiguous block of memory, reducing memory wastage compared to arrays, which need a fixed size.

**Limitations of Linked Lists:**

* **Access Time:** Accessing elements by index is slower (O(n)) compared to arrays (O(1)).
* **Memory Overhead:** Each node requires additional memory for storing the reference to the next node.