**Exercise 5: Task Management System**

**Step 1: Understand Linked Lists**

**Singly Linked List:**

* **Structure:** Each node contains data and a reference to the next node in the sequence.
* **Operations:**
  + **Insertion:** Efficient at the beginning; requires traversal to insert at the end.
  + **Deletion:** Efficient at the beginning; requires traversal to delete elsewhere.
  + **Traversal:** Only forward traversal is possible.

**Doubly Linked List:**

* **Structure:** Each node contains data, a reference to the next node, and a reference to the previous node.
* **Operations:**
  + **Insertion/Deletion:** Efficient at both ends; easier deletion as we have references to both the previous and next nodes.
  + **Traversal:** Can traverse both forward and backward.

**Step 4: Analysis**

**Time Complexity Analysis:**

* **Add Task:**
  + **Best Case:** O(1)O(1)O(1) if adding to the beginning.
  + **Worst Case:** O(n)O(n)O(n) if adding to the end.
* **Search Task:**
  + **Best Case:** O(1)O(1)O(1) if the task is at the beginning.
  + **Worst Case:** O(n)O(n)O(n) if the task is at the end or not present.
* **Delete Task:**
  + **Best Case:** O(1)O(1)O(1) if the task is at the beginning.
  + **Worst Case:** O(n)O(n)O(n) if the task is at the end or not present.
* **Traverse Tasks:** O(n)O(n)O(n), since all nodes are visited.

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

* **Dynamic Size:** Linked lists can grow or shrink as needed, unlike arrays with a fixed size.
* **Efficient Insertions/Deletions:** No need for shifting elements, unlike arrays.
* **Memory Usage:** More efficient in memory use for dynamic data, as arrays require pre-allocation of space.