**Exercise 5: Task Management System**

Q.Explain the different types of linked lists (Singly Linked List, Doubly Linked List).

A. **Singly linked list** is a linear data structure in which the elements are not stored in contiguous memory locations and each element is connected only to its next element using a pointer.

**A doubly linked list is a data structure consisting of a set of nodes, each containing** a **value** and **two pointers**, one pointing to the **previous node** in the list and one pointing to the**next node** in the list. This allows for efficient traversal of the list in**both directions**, making it suitable for applications where frequent **insertions** and **deletions** are required.

Q. Analyze the time complexity of each operation.

A. **Singly Linked List**:

* **Add**:
  + At the beginning: O(1)
  + At the end: O(n) (needs traversal to find the end)
  + At a specific position: O(n) (requires traversal to the position)
* **Search**: O(n) (requires traversal to find the element)
* **Traverse**: O(n) (visits each element once)
* **Delete**:
  + From the beginning: O(1)
  + From the end: O(n) (requires traversal to find the node before the last)
  + From a specific position: O(n) (requires traversal to the position)

**Doubly Linked List**:

* **Add**:
  + At the beginning: O(1)
  + At the end: O(1) (direct access to the end node)
  + At a specific position: O(n) (requires traversal to the position)
* **Search**: O(n) (requires traversal to find the element)
* **Traverse**: O(n) (visits each element once)
* **Delete**:
  + From the beginning: O(1)
  + From the end: O(1)
  + From a specific position: O(n) (requires traversal to the position)

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

A. **Dynamic Size**: Linked lists can grow or shrink in size dynamically, which is ideal for scenarios where the number of elements is not known in advance.

**Efficient Insertions/Deletions**: Linked lists allow efficient insertions and deletions from any position in the list without shifting elements, as opposed to arrays where elements need to be shifted.

**No Wasted Space**: Linked lists use memory proportional to the number of elements, avoiding wasted space associated with pre-allocated arrays.