1a.) **Types of Linked Lists**

1. **Singly Linked List**:
   * **Structure**: Each node contains data and a reference to the next node.
   * **Advantages**: Simple structure, efficient insertions and deletions (O(1) if you have a reference to the node).
   * **Disadvantages**: Only unidirectional traversal (from head to end).
2. **Doubly Linked List**:
   * **Structure**: Each node contains data, a reference to the next node, and a reference to the previous node.
   * **Advantages**: Bidirectional traversal (both forwards and backwards).
   * **Disadvantages**: Slightly more complex due to extra reference and requires more memory.

4a.) **Time Complexity**

1. **Add**:
   * **Time Complexity**: O(n) - Adding a task requires traversal to the end of the list.
   * **Space Complexity**: O(1) - Space is used for the new node.
2. **Search**:
   * **Time Complexity**: O(n) - Linear search is used to find a task by ID.
   * **Space Complexity**: O(1) - No extra space beyond the input list.
3. **Traverse**:
   * **Time Complexity**: O(n) - Traversing all tasks requires linear time.
   * **Space Complexity**: O(1) - No extra space beyond the input list.
4. **Delete**:
   * **Time Complexity**: O(n) - Linear search is used to find and remove a task, with potential traversal to the end of the list.
   * **Space Complexity**: O(1) - No extra space beyond the input list.

4b.) **Advantages of Linked Lists Over Arrays**

1. **Dynamic Size**:
   * **Advantage**: Linked lists can grow and shrink dynamically without a predefined size, unlike arrays.
2. **Efficient Insertions and Deletions**:
   * **Advantage**: Inserting or deleting elements in a linked list is efficient if you have a reference to the node (O(1) for insertion and deletion).
3. **No Waste of Space**:
   * **Advantage**: Unlike arrays, which may waste space if they are not fully utilized, linked lists use memory proportional to the number of elements.
4. **Flexibility**:
   * **Advantage**: Linked lists allow for easy implementation of more complex data structures like stacks and queues.