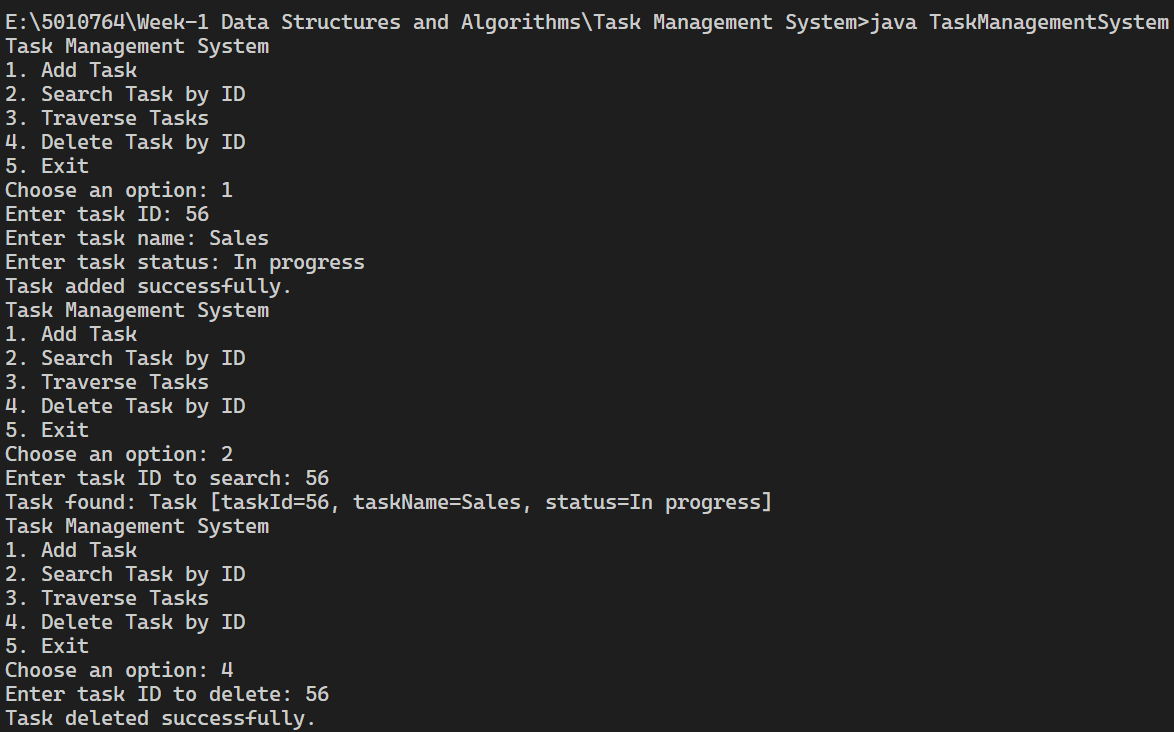
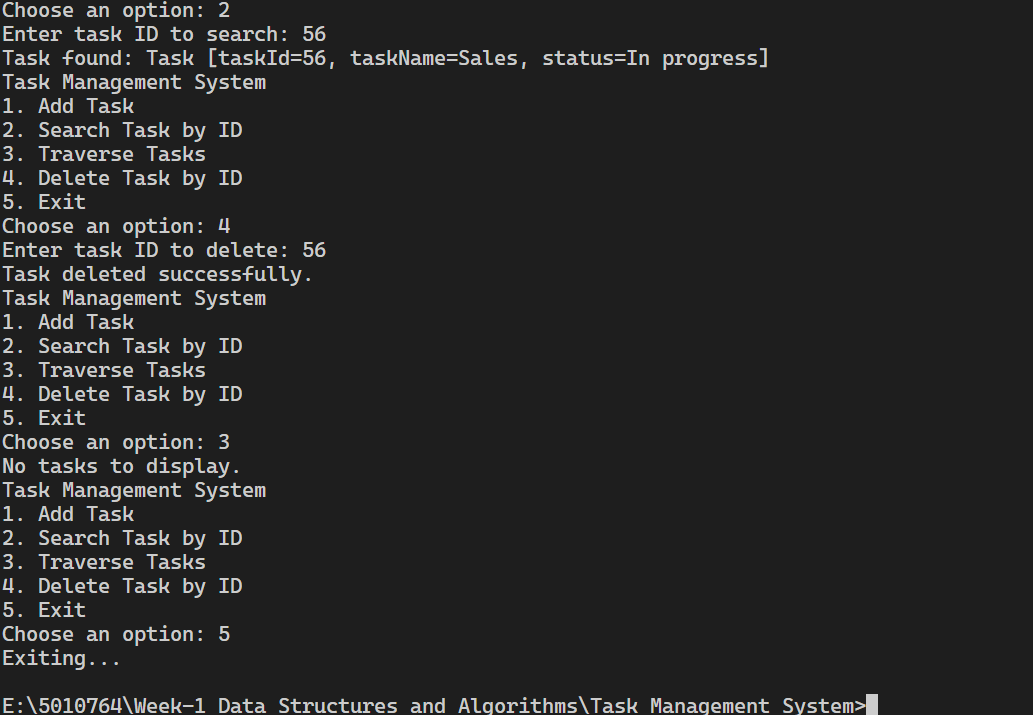
**TASK MANAGEMENT SYSTEM:**

1. **Understanding Linked Lists**
   * **Types of Linked Lists:**
     + **Singly Linked List: A data structure where each element (node) contains a value and a reference (link) to the next node in the sequence. It allows traversal in one direction, from the head to the end of the list.**
     + **Doubly Linked List: A more complex structure where each node contains a reference to both the next and the previous node, allowing traversal in both directions.**
2. **Setup**
   * **Define a Task class with attributes such as taskId, taskName, and status to represent each task.**
3. **Implementation**
   * **Singly Linked List Implementation: Use this structure to manage tasks, where each node in the list represents a Task.**
     + **Add Task: Create a new node and insert it at the appropriate position in the list.**
     + **Search Task: Traverse the list to find a node with a matching taskId.**
     + **Traverse Tasks: Iterate through the list to access each task's details.**
     + **Delete Task: Locate the node to be removed and adjust the links of adjacent nodes to exclude it.**
4. **Analysis**
   * **Time Complexity Analysis:**
     + **Add Task: O(1) if adding at the head, or O(n) if inserting at a specific position.**
     + **Search Task: O(n) as it requires traversing the list to find the node.**
     + **Traverse Tasks: O(n) as each node must be visited.**
     + **Delete Task: O(n) due to the need to locate the node and adjust pointers.**
   * **Advantages of Linked Lists Over Arrays for Dynamic Data:**
     + **Dynamic Size: Linked lists can grow or shrink in size dynamically without the need for resizing, unlike arrays.**
     + **Efficient Insertions/Deletions: Inserting or deleting nodes can be done more efficiently in a linked list without shifting elements, as required in arrays.**
5. **Output**

****

****