**Exercise 5: Task Management System**

**1.Understand Linked Lists:**

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

A [linked list](https://www.geeksforgeeks.org/data-structures/linked-list/) is a linear data structure, in which the elements are not stored at contiguous memory locations. The elements in a linked list are linked using [pointers](https://www.geeksforgeeks.org/pointers-in-c-and-c-set-1-introduction-arithmetic-and-array/). In simple words, a linked list consists of nodes where each node contains a data field and a reference(link) to the next node in the list.

***Singly Linked List:***

*A singly linked list is a fundamental data structure in computer science and programming, it consists of nodes where each node contains a data field and a reference to the next node in the node. The last node points to null, indicating the end of the list. This linear structure supports efficient insertion and deletion operations, making it widely used in various applications. In this tutorial, we’ll explore the node structure, understand the operations on singly linked lists (traversal, searching, length determination, insertion, and deletion), and provide detailed explanations and code examples to implement these operations effectively.*

***Doubly Linked List:***

A doubly linked list is a more complex data structure than a singly linked list, but it offers several advantages. The main advantage of a doubly linked list is that it allows for efficient traversal of the list in both directions. This is because each node in the list contains a pointer to the previous node and a pointer to the next node. This allows for quick and easy insertion and deletion of nodes from the list, as well as efficient traversal of the list in both directions.

**4. Analysis:**

***o Analyze the time complexity of each operation.***

**Add Task**: Adding a new task to the end of the linked list.

Time Complexity:

Best Case: O(1) (if the list is empty)

Worst Case: O(n) (if the list has n elements)

**Search Task**: Searching for a task by taskId.

Time Complexity:

Best Case: O(1) (if the task is at the head)

Worst Case: O(n) (if the task is at the end or not present)

**Traverse Tasks**: Traversing and printing all tasks in the linked list.

Time Complexity: O(n) (as each node must be visited once)

**Delete Task:** Deleting a task by taskId.

Time Complexity:

Best Case: O(1) (if the task to delete is at the head)

Worst Case: O(n) (if the task is at the end or not present)

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

* **Dynamic Size:** Linked lists can grow and shrink dynamically as needed, while arrays have a fixed size.
* **Efficient Insertions/Deletions:** Insertions and deletions are more efficient in linked lists, especially at the beginning or middle of the list, compared to arrays where shifting elements is required.
* **Memory Utilization:** Linked lists use memory more efficiently for dynamic data as they allocate memory as needed.