**Exercise 5: Task Management System**

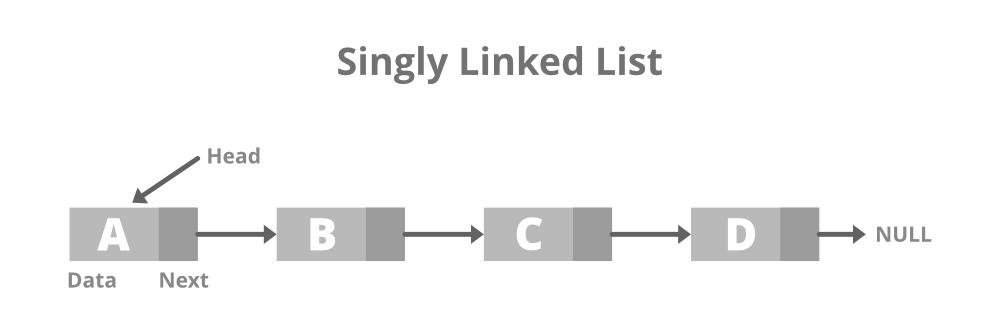
**Scenario:**

You are developing a task management system where tasks need to be added, deleted, and traversed efficiently.

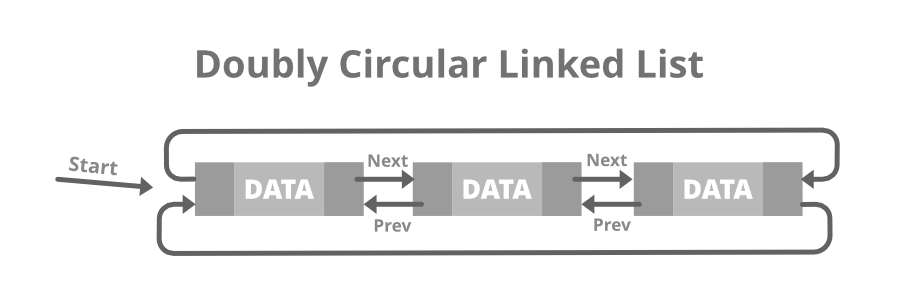
**Steps:**

1. **Understand Linked Lists:**
   * Explain the different types of linked lists (Singly Linked List, Doubly Linked List).

**Singly Linked List**: A **singly linked list** is a set of nodes where each node has two fields ‘data’ and ‘next address. The ‘data’ field stores actual piece of information and ‘link’ field is used to point to next node. Basically the ‘next address’ field stores the address of the next node.



**Doubly Linked List**: A **Doubly Linked List** contains an extra pointer, typically called *previous pointer*, together with next pointer and data which are there in singly linked list.



1. **Analysis:**
   * Analyze the time complexity of each operation.

Time Complexity:

* **Add**: O(n), because we need to traverse the linked list to find the last node.
* **Search**: O(n), because we need to traverse the linked list to find the task with the given ID.
* **Traverse**: O(n), because we need to traverse the entire linked list to print all tasks.
* **Delete**: O(n), because we need to traverse the linked list to find the task with the given ID and then delete it.
  + Discuss the advantages of linked lists over arrays for dynamic data.

**Advantages of a Linked List over Arrays**

1. **Dynamic Size:** One of the most significant advantages of linked list over arrays is that linked lists can grow or shrink dynamically during runtime. This means that the size of a linked list can be adjusted to accommodate new elements or remove existing elements without having to allocate or deallocate a fixed-size block of memory, as is the case with arrays.
2. **Efficient Insertion and Deletion:** Linked lists allow efficient insertion and deletion of elements at any position in the list, whereas arrays require shifting of elements when a new element is added or removed, which can be slow and inefficient for large arrays.
3. **Memory Efficiency:** Linked lists use memory more efficiently than arrays. In an array, all elements must be stored in contiguous memory locations, even if some of the elements are not used. In contrast, linked lists only allocate memory for the elements that are used, which can save memory in cases where the size of the data set is unknown or varies over time.
4. **More Efficient Sorting:** In some cases, linked lists can be more efficient for sorting algorithms than arrays. This is because linked lists do not require swapping elements like arrays, which can be time-consuming for large arrays.