Question:

What are the differences between **Singly Linked List**, **Circular Linked List**, and **Doubly Linked List** in terms of their **uses**, **advantages (pros)**, and **disadvantages (cons)**?

Answer:

1. Singly Linked List

A singly linked list is a collection of nodes where each node contains data and a reference to the next node in the list.

Uses:

- Used when memory usage is important.
- Suitable for implementing stacks and queues.
- Useful when traversal is needed in only one direction.

Pros:

- Requires less memory compared to doubly linked lists.
- Simple and easy to implement.
- Dynamic size (can grow or shrink easily).

Cons:

- Traversal is only possible in one direction.
- Deletion of a node requires access to the previous node.
- Searching operations are slower compared to arrays.

---

2. Circular Linked List

A circular linked list is a linked list where the last node points back to the first node instead of pointing to `null`.

Uses:

- Useful in applications that require continuous looping, such as round-robin scheduling.
- Used in circular queues.
- Suitable for applications where the starting point is not fixed.

Pros:

- No node points to `null`, allowing continuous traversal.
- Efficient use of memory compared to doubly linked lists.
- Any node can be treated as the first node.

Cons:

- More complex to implement and manage.
- Traversal can result in infinite loops if not handled carefully.
- Debugging is harder than singly linked lists.

---

3. Doubly Linked List

A doubly linked list is a collection of nodes where each node contains data and two references: one to the previous node and one to the next node.

Uses:

- Used in navigation systems such as browser history.
- Suitable for implementing deque (double-ended queue).
- Useful when bidirectional traversal is required.

Pros:

- Allows traversal in both forward and backward directions.
- Easier deletion of nodes since previous node is directly accessible.
- More flexible for complex data manipulation.

Cons:

- Requires more memory due to storing two references per node.
- More complex to implement compared to singly linked lists.
- Slightly slower due to extra pointer operations.