LAB 6
Implementation of Linked List II

NOTE: Perform all tasks using Class template. Handle All cases for both Insertion & Deletion at Start, Middle and end. Show Output Screenshots at the end of Insertion, Deletion, Searching & Traversing.

Task 1: Write a menu driven program for Sorted Insertion, Deletion, Traversing, Searching a Doubly Linked list.

Hint:
The node is defined inside the `DoublyLinkedList` class as a private structure (`struct Node`), ensuring that the node's details are encapsulated.
The `prev` pointer is crucial for enabling backward traversal, while the `next` pointer helps in forward traversal.
Use conditional checks to handle three main cases:

- Empty list.
- Insert at the beginning (new head).
- Insert somewhere in the middle or end.

Node Deletion: Handle the following cases separately:

- Deleting the head node.
- Deleting a node in the middle or at the end.
- Make sure to update the pointers (`prev` and `next`) correctly to avoid dangling references.

Traversal (Forward and Backward):
For backward traversal, first locate the last node by iterating through the list, then traverse backward by following the `prev` pointers.

Searching for a Node: Keep track of the position using a counter, and compare each node's `data` with the value you are searching for.

Use a `while` loop to delete each node from the head until the list becomes empty (`head == nullptr`).