Lab 3: Doubly Linked List Operations (by Group)

Objective: Implement and test operations on a **doubly linked list** using the given program structure.

Submission : Individual + give marks for your members . C++ program .

Score : If you decide to complete it on your own, with or without the help of AI tools, please provide your own score (Example 7/10)

Instructions

1. **Understand the Given Code Structure**:
   - **Node Class**: Represents a single node in the doubly linked list with name, prev, and next attributes.
   - **DoublyLinkedList Class**: Provides methods to perform operations on the list.
2. **Complete the Tasks**:
   - Follow the hints and implement the missing parts of the program step by step.
3. **Tasks to Implement**:

   Task 1: Complete insertAtEnd

   - This method appends a new node with the given name at the end of the doubly linked list.
   - **Hint**:
     - If the list is empty (head is nullptr), initialize the list with the new node.
     - Otherwise, traverse to the tail node and link the new node after it.

   Task 2: Complete countAndDisplay

   - This method counts the total number of nodes in the list and displays each node's name.
   - **Hint**: Use a loop to traverse the list and keep a count of nodes.

   Task 3: Complete deleteLastNode

   - This method removes the last node from the list.
   - **Hint**:
     - If the list is empty, display an appropriate message.
     - If there's only one node, update both head and tail to nullptr.
     - Otherwise, unlink the last node and update the tail.

   Task 4: Complete insertAtSecond

   - This method inserts a new node at the second position in the list.
   - **Hint**:
     - If the list has less than two nodes, handle appropriately.

        ▪ Otherwise, adjust the next and prev pointers to insert the new node.

Task 5: Complete displayList

- This method prints the names of all nodes in the list in order.
- **Hint**: Traverse the list starting from head and display each node's name.

4. **Compile and Run the Program**

5. **Test Your Implementation**:

- Ensure the program correctly performs the following:
  - **Step 1**: Create a list with names: `"Ali"`, `"Baba"`, `"Chan"`, `"Diana"`, `"Ely"`.
  - **Step 2**: Count and display the nodes.
  - **Step 3**: Delete the last node (`"Ely"`) and display the updated list.
  - **Step 4**: Insert a new node (`"Alisa"`) at the second position and display the updated list.

  Sample output :

```
Node 1: Ali
Node 2: Baba
Node 3: Chan
Node 4: Diana
Node 5: Ely

Total Nodes: 5



List after deleting last node:

Node 1: Ali
Node 2: Baba
Node 3: Chan
Node 4: Diana



List after inserting 'Alisa' at second position:

Node 1: Ali
Node 2: Alisa
Node 3: Baba
Node 4: Chan
Node 5: Diana
```