Lab Assignment: Find the Middle of a Singly Linked List

Problem Statement:
Given a singly linked list, write a function to find and return the middle element of the list. In case of an even number of nodes, return the *second middle node*. Students are required to **implement the optimized strategy**.

Discussion of Possible Approaches:

1. **Naïve Approach (Two-Pass):**
   - First traverse the list to count the total number of nodes.
   - Then traverse again to the node at position $\lceil n/2 \rceil$ (second middle if n is even).
   - Time Complexity: $O(n)$, Space Complexity: $O(1)$.
2. **Optimized Approach (Fast-Slow Pointers):**
   - Maintain two pointers: `slow` and `fast`.
   - Move `slow` one step at a time and `fast` two steps at a time.
   - When `fast` reaches the end, `slow` will be at the middle node.
   - This works in a single traversal.
   - Time Complexity: $O(n)$, Space Complexity: $O(1)$.

Function Signature to Implement:

```
int getMiddle(Node* head);
```

Examples:

- Input: `1 -> 2 -> 3 -> 4 -> 5` Output: `3` (5 nodes, middle is node 3)
- Input: `10 -> 20 -> 30 -> 40 -> 50 -> 60` Output: `40` (6 nodes, two middle nodes are 30 and 40; return the second: 40)