Linked List Cycle

For this problem the goal is to detect if a linked list has a cycle, and the optimal approach is to use Floyd's Cycle Detection Algorithm (Tortoise and Hare algorithm).

Approach: Floyd's cycle detection

Idea:

- Use two pointers:
  - Slow pointer moves 1 step at a time.
  - Fast pointer moves 2 steps at a time.
- If there is a cycle, slow and fast will eventually meet.
- If fast or fast->next becomes nullptr, there is no cycle.

Steps:

1. Initialize slow = head, fast = head.
2. Move slow by 1 step and fast by 2 steps in each iteration.
3. If slow == fast at any point, return true (cycle exists).
4. If fast or fast->next is nullptr, return false (no cycle).

Complexity Analysis:

- Time Complexity: O(n) – Each node is visited at most twice.
- Space Complexity: O(1) – No extra memory used.