Lab Assignment: Detect Loop in a Linked List

Objective

Implement a function in C to detect whether a given singly linked list contains a cycle (loop) using the Floyd's Cycle-Finding algorithm (also known as the Tortoise and Hare algorithm).

Problem Statement

Given the head of a singly linked list, determine if there exists a loop in the list. Return 1 (or true) if a loop exists, otherwise return 0 (or false). Use **O(n)** time complexity and **O(1)** auxiliary space.

Function Signature

```
int detectLoop(struct Node* head);
```

Approach (Floyd's Algorithm)

- Initialize two pointers, `slow` and `fast`, both pointing to `head`.
- Move `slow` by one node and `fast` by two nodes in each iteration.
- If at any point `slow == fast`, a loop exists—return 1.
- Continue until `fast` or `fast->next` becomes `NULL`, indicating no loop—return 0.

Example Cases

- **Loop Present:** Input: `1 -> 2 -> 3 -> 4 -> 5 -> 2 (loop back)` Output: 1 (true)
- **No Loop:** Input: `10 -> 20 -> 30 -> 40 -> 50` Output: 0 (false)