Detect and Remove Loop from Linked List CodeStudio

You are given a singly linked list. Write a C++ program that detects and removes any cycles from the linked list.

Example:

1 -> 2 -> 3 -> 4 -> 5

|_____|

Output:

1 -> 2 -> 3 -> 4 -> 5 -> NULL

Approach 1: Hash Set (unordered_map) for Cycle Detection

The hash map approach utilizes a hash map (unordered_map in C++) to track visited nodes while traversing the linked list. This approach consists of the following steps:

- Initialize a hash map to store visited nodes and a pointer (**currNode**) to traverse the linked list.
- Traverse the linked list:
    - If the current node is already present in the hash map, a cycle is detected, and the function returns the node where the cycle starts.
    - Otherwise, mark the current node as visited by adding it to the hash map and move to the next node.
- **Time Complexity: O(n), where n is the number of nodes in the linked list.**
- **Space Complexity: O(n), due to the space required by the hash map to store visited nodes.**

Approach 2: Floyd's Cycle Detection Algorithm (Tortoise and Hare)

Floyd's Cycle Detection Algorithm, also known as the Tortoise and Hare algorithm, uses two pointers to traverse the linked list. This approach involves the following steps:

- Initialize two pointers (**slow** and **fast**) to the head of the linked list.
- While traversing the linked list:
    - Move the **slow** pointer one step at a time and the **fast** pointer two steps at a time.
    - If the pointers meet at the same node, a cycle is detected, and the loop terminates.

- After detecting the cycle, reset the **slow** pointer to the head of the linked list.
- Traverse the linked list again using both the **slow** and **fast** pointers, moving them one step at a time.
    - When the two pointers meet again, the meeting point is the start of the cycle.
- Break the cycle by setting the **next** of the last node in the cycle to **nullptr**.
- **Time Complexity: O(n), where n is the number of nodes in the linked list.**
- **Space Complexity: O(1), as only a constant amount of extra space is used.**

Function to remove a detected loop from the linked list.

The **removeLoop** function combines both cycle detection approaches and employs the detected starting node of the cycle to remove the cycle from the linked list. It uses the selected approach (hash map or Floyd's algorithm) to identify and remove the cycle, ultimately returning the head of the linked list without any cycles.