Here's how we can solve the problem step by step:

Approach: Floyd's Cycle Detection Algorithm

Detect if a loop exists

1. Use two pointers:

2 Slow Pointer moves one step at a time.

3 Fast Pointer moves two steps at a time.

If they meet, a loop exists.

Find the starting node of the loop

Reset the Slow Pointer to the head of the linked list.

Move both Slow Pointer and Fast Pointer one step at a time.

The node where they meet is the starting node of the loop.

If no loop exists, return -1.

```
#include <stdio.h>
#include <stdlib.h>

struct Node {
int data;
struct Node* next;
};
```

```
// Function to detect and return the start of the loop
struct Node* detectLoop(struct Node* head) {
struct Node *slow = head, *fast = head;

// Step 1: Detect loop using Floyd's algorithm
while (fast != NULL && fast->next != NULL) {
slow = slow->next;
fast = fast->next->next;

if (slow == fast) { // Loop detected
slow = head;

// Step 2: Find the start of the loop
while (slow != fast) {
slow = slow->next;
fast = fast->next;
}
return slow; // Loop start node
}
}
return NULL; // No loop
}

// Function to create a new node
struct Node* newNode(int key) {
struct Node* temp = (struct Node*)malloc(sizeof(struct Node));
temp->data = key;
temp->next = NULL;
return temp;
}
```

```
int main() {
struct Node* head = newNode(1);
head->next = newNode(2);
head->next->next = newNode(3);
head->next->next->next = newNode(4);
head->next->next->next->next = newNode(5);

// Creating a loop for testing
head->next->next->next->next->next = head->next->next; // Loop at node 3

struct Node* loopNode = detectLoop(head);
if (loopNode)
printf("Loop starts at node with value: %d\n", loopNode->data);
else
printf("No loop detected\n");

return 0;
}
```