class Solution {

public:

ListNode\* reverseBetween(ListNode\* head, int left, int right) {

if (!head || left == right) return head; // Edge case: No reversal needed

// Dummy node to handle cases where left = 1

ListNode\* dummy = new ListNode(0);

dummy->next = head;

ListNode\* prev = dummy;

// Move prev to the node before left

for (int i = 1; i < left; i++)

prev = prev->next;

// Reverse the sublist from left to right

ListNode\* curr = prev->next;

ListNode\* nextNode = nullptr;

ListNode\* prevNode = nullptr;

for (int i = left; i <= right; i++) {

nextNode = curr->next;

curr->next = prevNode;

prevNode = curr;

curr = nextNode;

}

// Connect reversed sublist with remaining part of the list

prev->next->next = curr; // Connect end of reversed sublist to nextNode

prev->next = prevNode; // Connect prev to start of reversed sublist

ListNode\* newHead = dummy->next;

delete dummy;

return newHead;

}

}; }

return false; // No cycle found

}

};