Detect Loop in linked list

Easy Accuracy: 43.49% Submissions: 339K+ Points: 2

Given a linked list of **N** nodes. The task is to check if the linked list has a loop. Linked list can contain self loop.

Example 1:

Input:
N = 3
value[] = {1,3,4}
x(position at which tail is connected) = 2
Output: True
Explanation: In above test case N = 3. The linked list with nodes N = 3 is given. Then value of x=2 is given which means last node is connected with xth node of linked list. Therefore, there exists a loop.

C++ (g++ 5.4) Average Time: 20m Start Timer

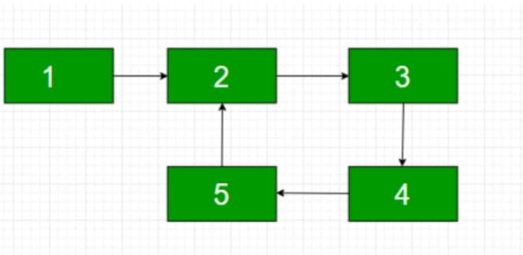
```
// } Driver Code Ends
//User function template for C++


class Solution
{
    public:
    //Function to check if the linked list has a loop.
    bool detectLoop(Node* head)
    {
        Node* curr = head;
        if(head == NULL || head->next == NULL){
            return false;
        }
        Node *slow = head;
        Node *fast = head;
        while(fast != NULL && fast->next != NULL){
            slow = slow->next;
            fast = fast->next->next;
            if(slow == fast){
                return true;
            }
        }
        return false;
    }
};

// } Driver Code Ends
```

Custom Input | Compile & Run | Submit

Find length of Loop

Easy Accuracy: 44.26% Submissions: 102K+ Points: 2

Given a linked list of size **N**. The task is to complete the function **countNodesinLoop()** that checks whether a given Linked List contains a **loop or not** and if the **loop** is present then **return the count of nodes** in a loop or else **return 0. C** is the position of the node to which the last node is connected. If it is 0 then no loop.



C++ (g++ 5.4) Average Time: 30m Start Timer

```
1  // } Driver Code Ends
61
62
63 int countNodesinLoop(struct Node *head)
64 {
65     if(head->next == NULL && head == NULL){
66         return 0;
67     }
68     Node *slow = head->next;
69     Node *fast = head->next->next;
70     while(slow != fast){
71         if(fast == NULL || fast->next == NULL){
72             return 0;
73         }
74         slow = slow->next;
75         fast = fast->next->next;
76     }
77     fast = fast->next;
78     int cnt= 1;
79     while(slow != fast){
80         cnt++;
81         fast = fast->next;
82     }
83     return cnt;
84
85 }
```

Custom Input | Compile & Run | Submit