Newton School of Technology

20

Linked List – Part III

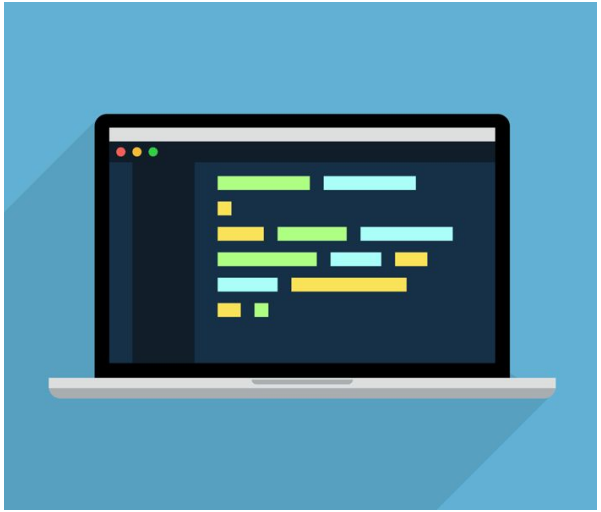
by Gladden Rumao

CSA 221 : DSA

Newton School of Technology

Quick Recap :

- Two Pointers Approach
- Hare and Tortoise Method
- Middle of Linked List
- Floyd’s Cycle Detection

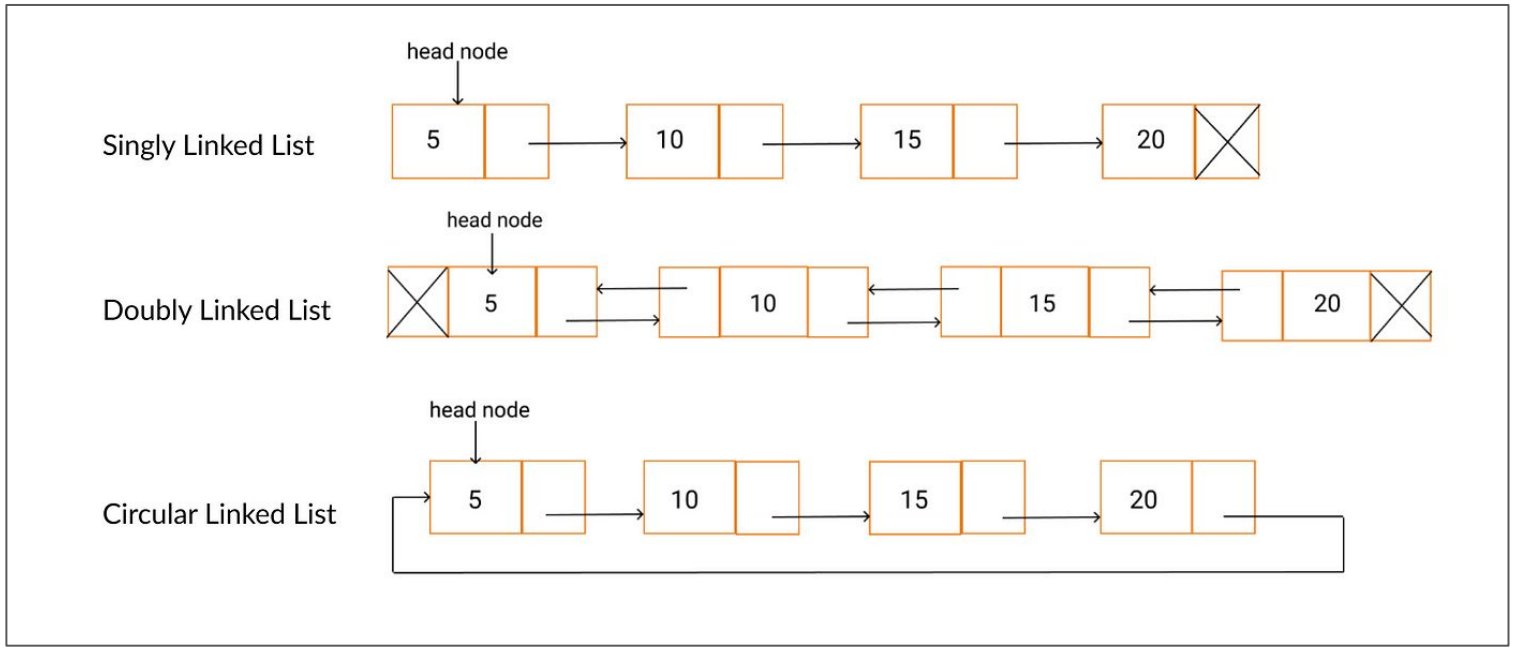


Newton School of Technology

Types of Linked List

Newton School of Technology

Types of Linked List :



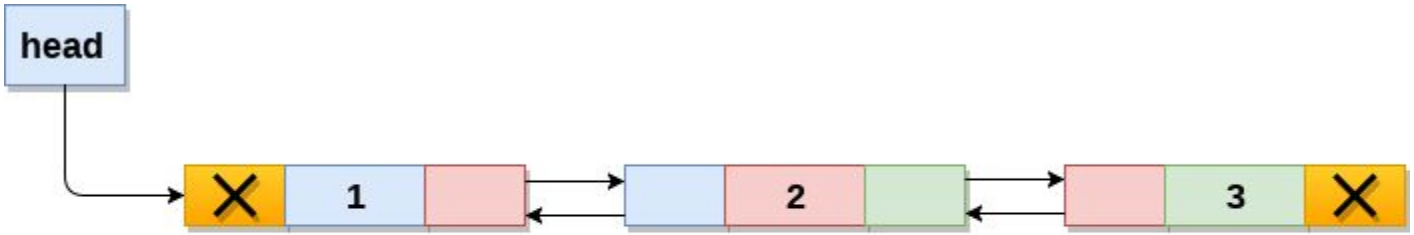
Newton School of Technology

Doubly Linked List

Newton School of Technology

Doubly Linked List :

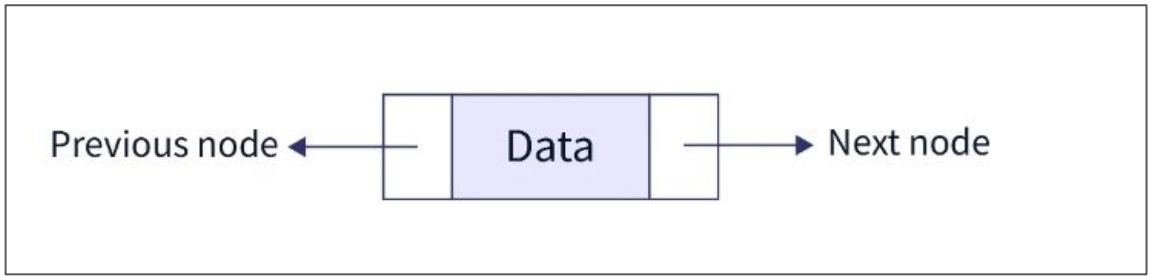
A doubly linked list (DLL) is a special type of linked list in which each node contains a pointer to the previous node as well as the next node of the linked list.



Newton School of Technology

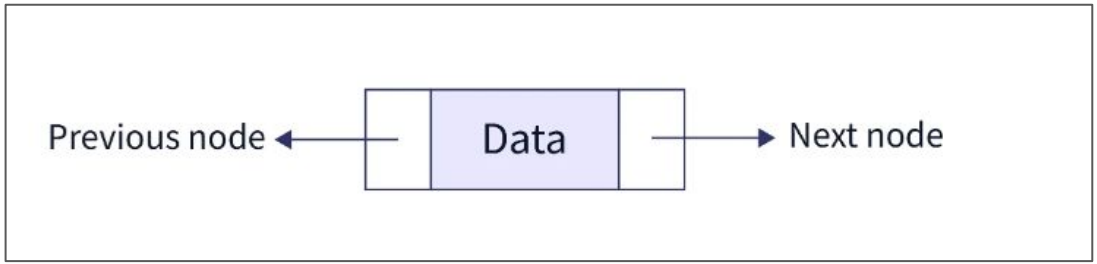
Node Structure of DLL :

```
class Node:
    def __init__(self, data):
        self.data = data
        self.prev = None
        self.next = None
```



Newton School of Technology

Node Structure of DLL :



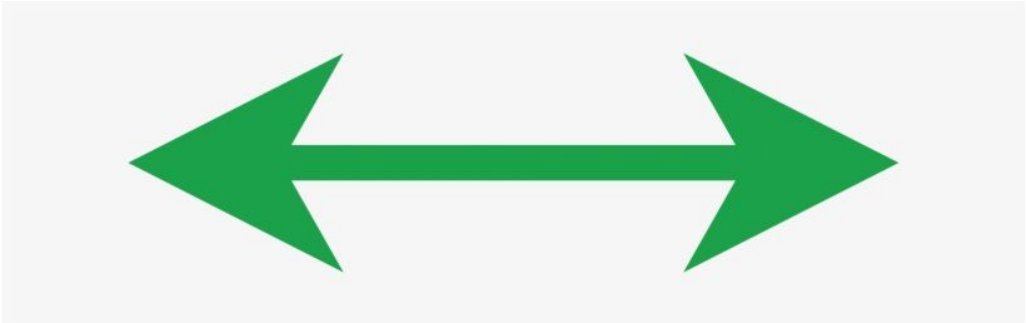
A **Doubly Linked List** is a linked list where each node has **two pointers**:

1. `prev` → Points to the **previous node**
2. `next` → Points to the **next node**

Newton School of Technology

Important Points :

- Unlike **singly linked list**, it allows **bi-directional traversal**.
- More efficient **insertion & deletion** than singly linked list.
- Uses **extra memory** for `prev` pointer.



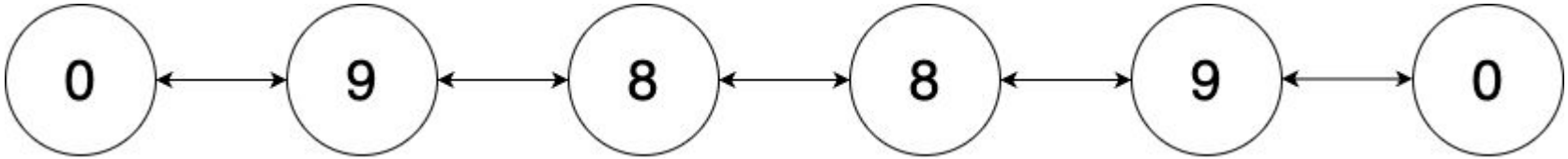
Newton School of Technology

Check Palindrome - Doubly Linked List

Newton School of Technology

Problem Statement :

Given head of a doubly linked list, your task is to check whether the linked list is a palindrome or not.



Newton School of Technology

How will you solve the problem ?

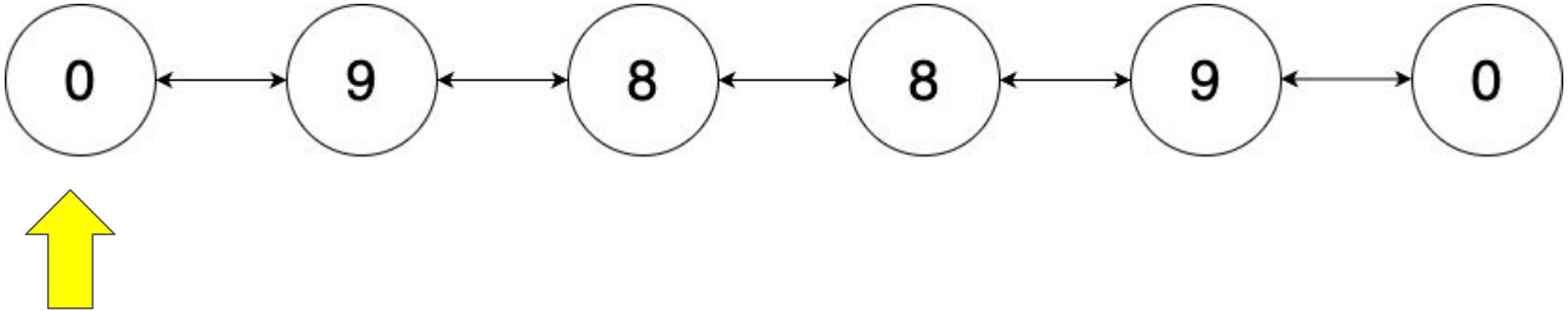
Newton School of Technology

Intuition

1. **Find Last Node:** Iterate till the last node
2. **Apply 2 Pointer technique:**
    - First Pointer points to head node
    - Second Pointer points to Last node
3. **Check whether first and last node data is equal or not**
    - *It is is equal update the pointers.*
        i. First will move forward
        ii. Second will move backwards
    - *If not equal directly return false (linked list is not palindrome)*
4. **After checking for all values we will return true** (means palindrome)

Newton School of Technology

Palindrome in Doubly Linked List:

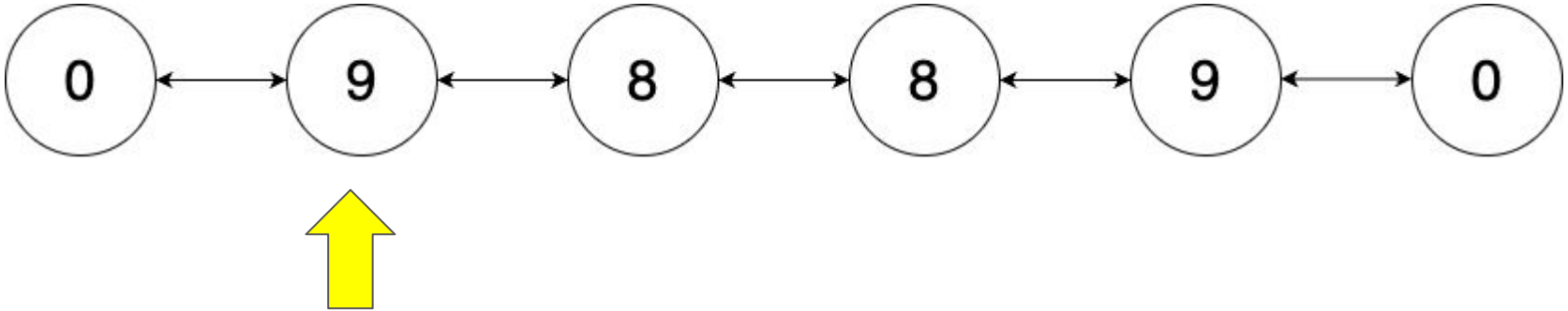


Node next is not Null

Jump to next node

Newton School of Technology

Palindrome in Doubly Linked List:

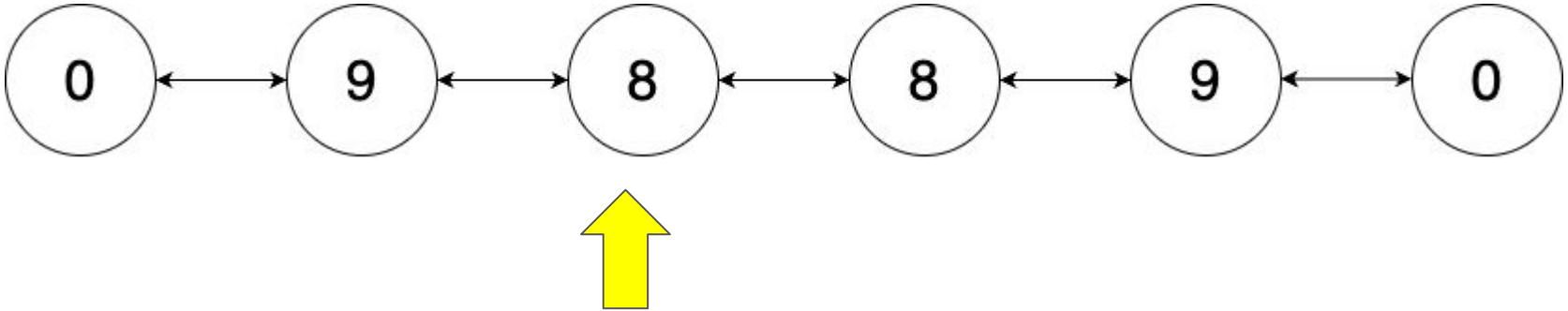


Node next is not Null

Jump to next node

Newton School of Technology

Palindrome in Doubly Linked List:

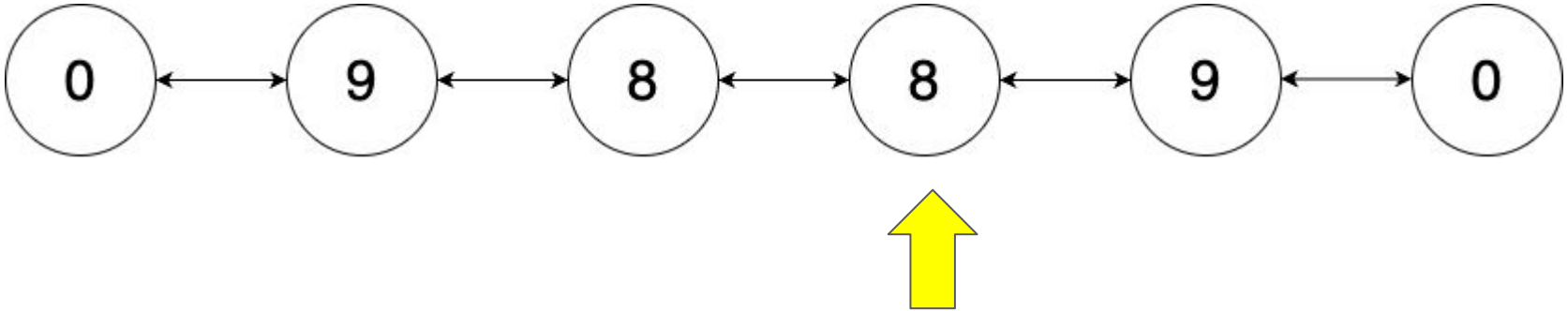


Node next is not Null

Jump to next node

Newton School of Technology

Palindrome in Doubly Linked List:

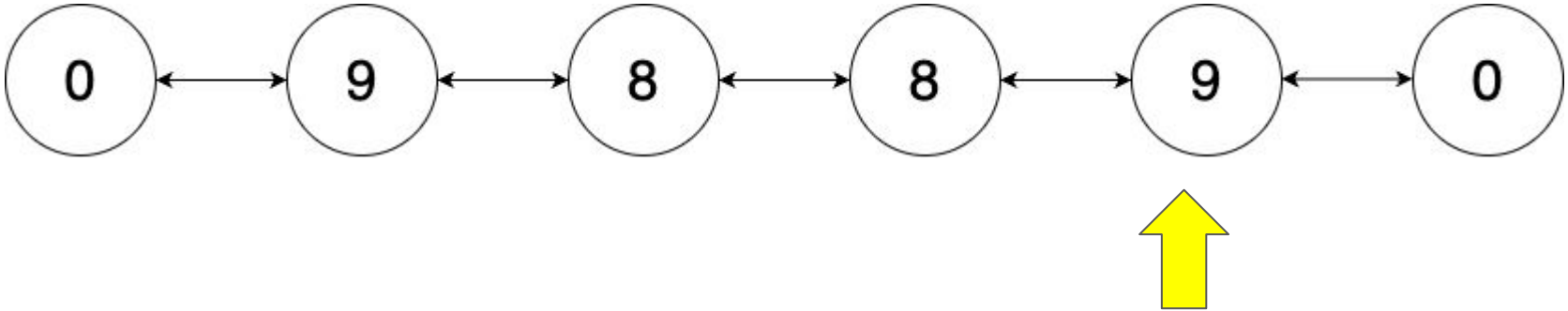


Node next is not Null

Jump to next node

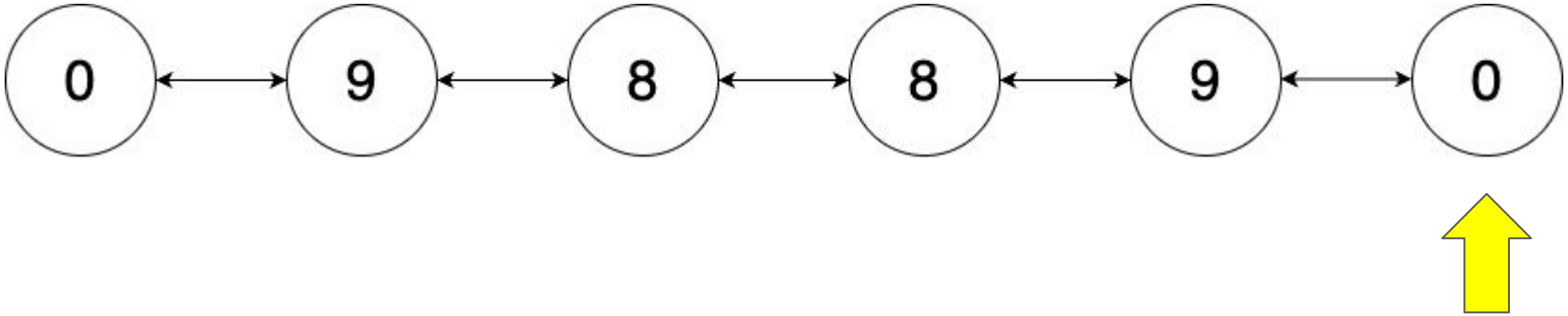
Newton School of Technology

Palindrome in Doubly Linked List:



Newton School of Technology

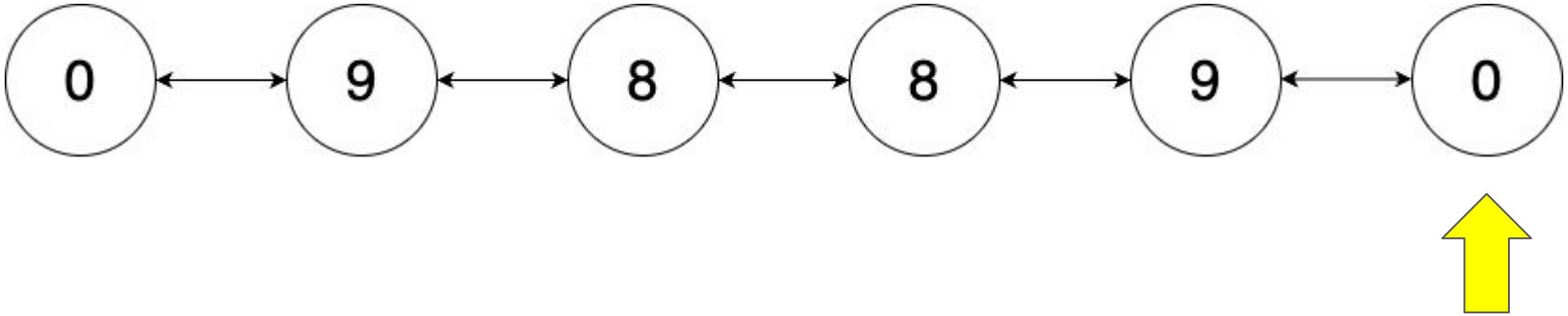
Palindrome in Doubly Linked List:



Node next is Null

(Last Node)

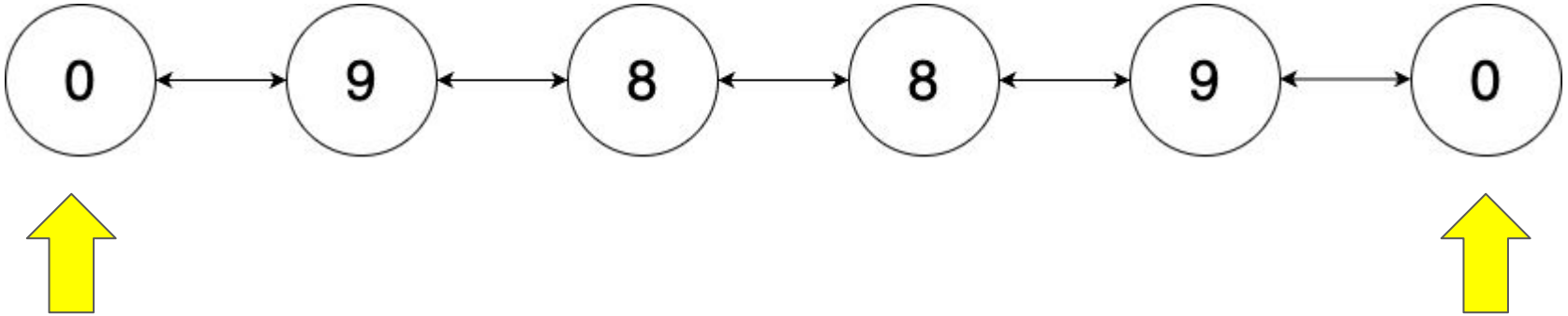
Palindrome in Doubly Linked List:



1. Apply Two Pointer Technique
2. One Pointer at start, Second Pointer at End
3. Compare the values of both pointers
4. If Values equal then check for remaining nodes

Newton School of Technology

Palindrome in Doubly Linked List:

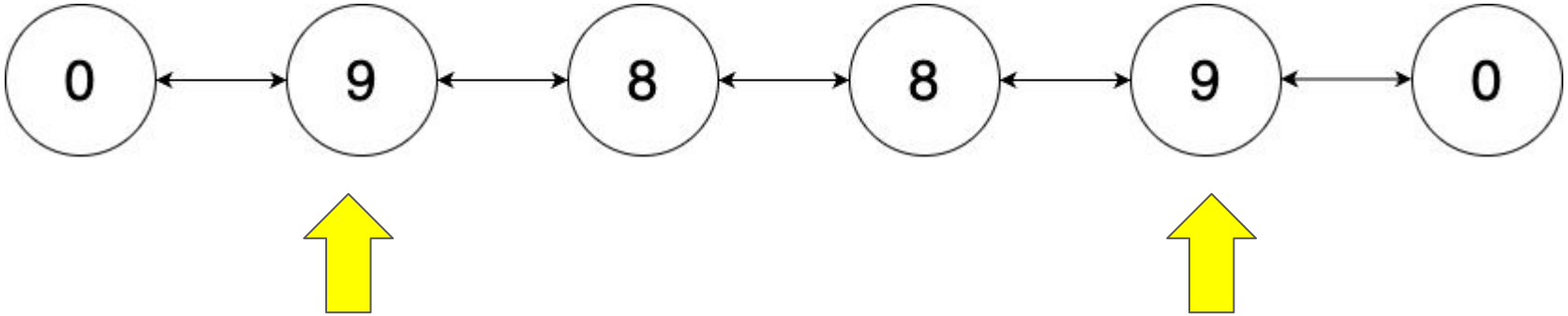


Both values equal

Update the pointers

Newton School of Technology

Palindrome in Doubly Linked List:

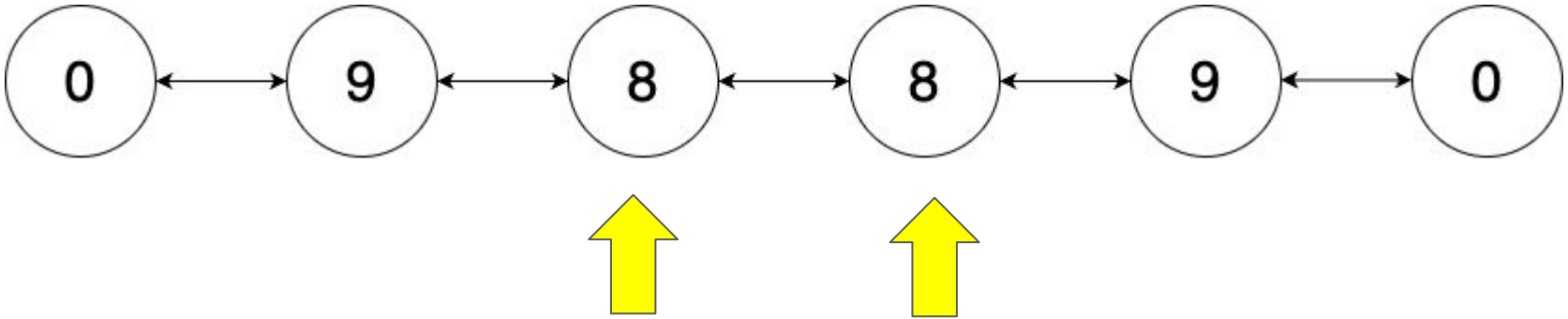


Both values equal

Update the pointers

Newton School of Technology

Palindrome in Doubly Linked List:

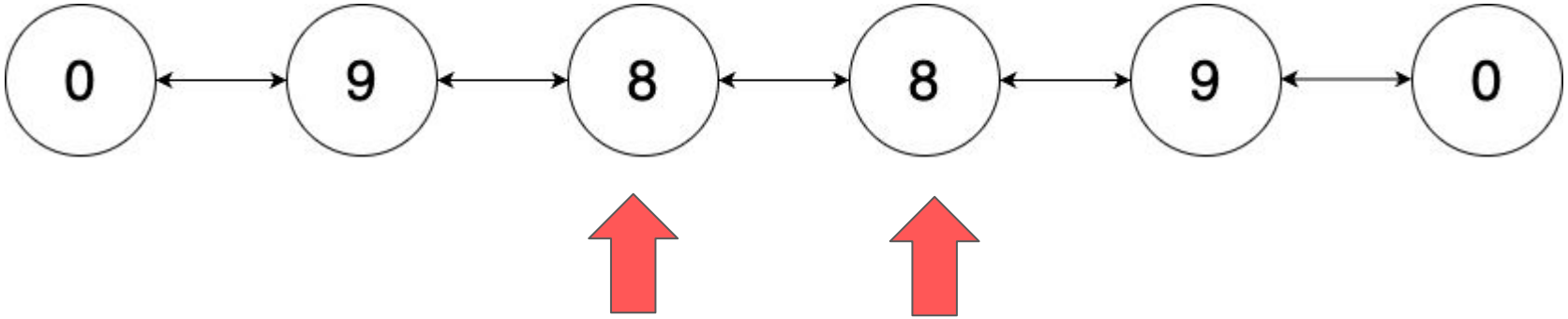


Both values equal

Update the pointers

Newton School of Technology

Palindrome in Doubly Linked List:

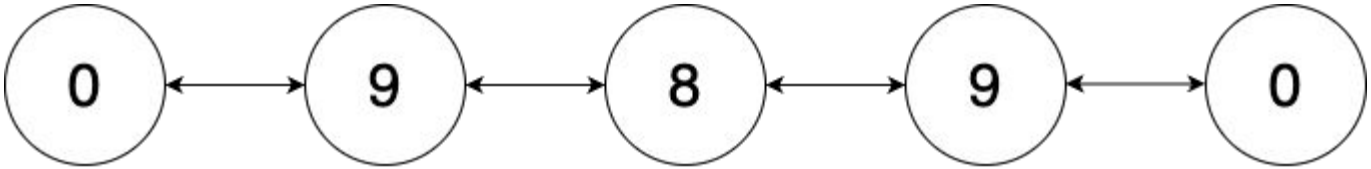


Pointers cross each other

It means first half is equal to second half

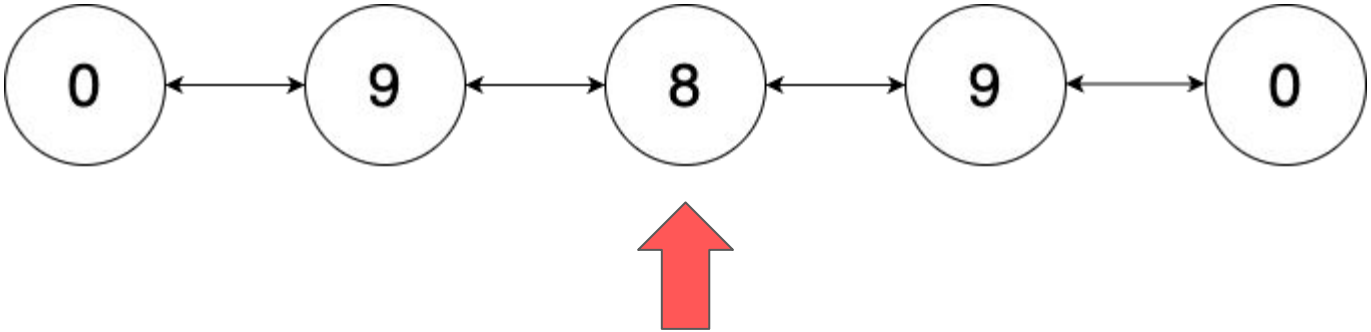
Linked List is palindrome

Palindrome in Doubly Linked List:



How will you take care for odd length linked list ?

Palindrome in Doubly Linked List:



For odd length when both pointers meet at same node we gonna stop it.

Linked List is palindrome

Newton School of Technology

Code :

```
def isPalin(left):
    if left is None:
        return True

    right = left
    while right.next is not None:
        right = right.next

    while left != right:
        if left.data != right.data:
            return False

        left = left.next
        right = right.prev

    return True
```

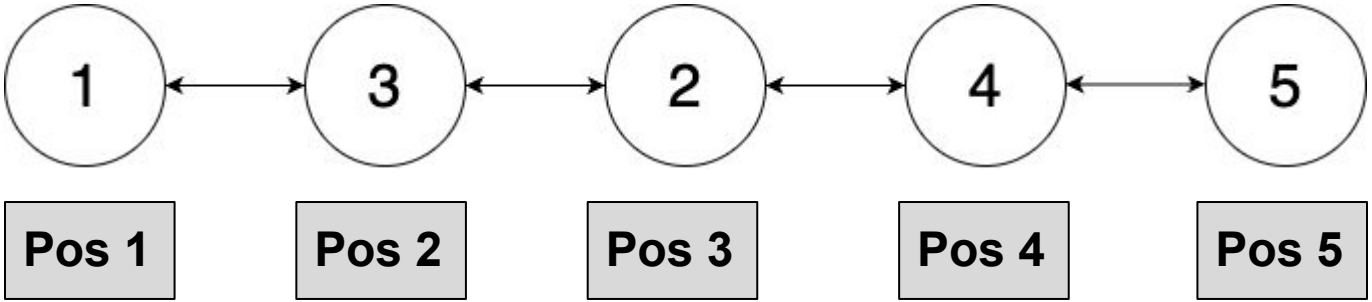
Newton School of Technology

Question - Insert in Doubly Linked List

Newton School of Technology

Problem Statement :

Given a doubly linked list consisting of N nodes and two integers val and pos. Your task is to add an element val at the pos position from the start of the linked list

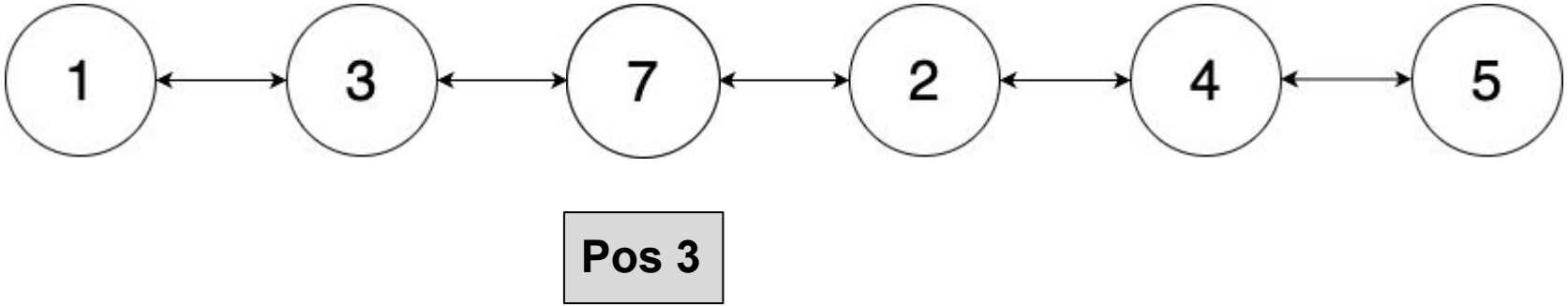


Insert value 7 at Position 3

Newton School of Technology

Resultant:

Value 7 is added at position 3



Newton School of Technology

How will you solve the problem ?

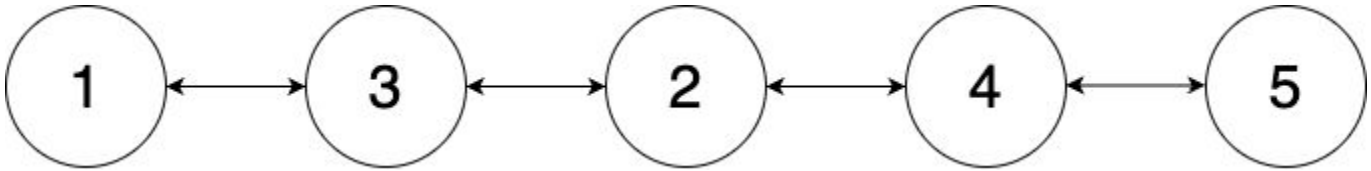
Newton School of Technology

Intuition

1. **Find the pos-1 node**
   - To update pointer of desired node
2. **After locating that node**
   - Insert new_node to it's next position which is 'pos' position
3. **Follow these 4 steps**
   - new_node.prev = cur
   - new_node.next = cur.next
   - cur.next = new_node
   - new_node.next.prev = new_node
4. **After inserting the new_node**
   - Return the **head** node

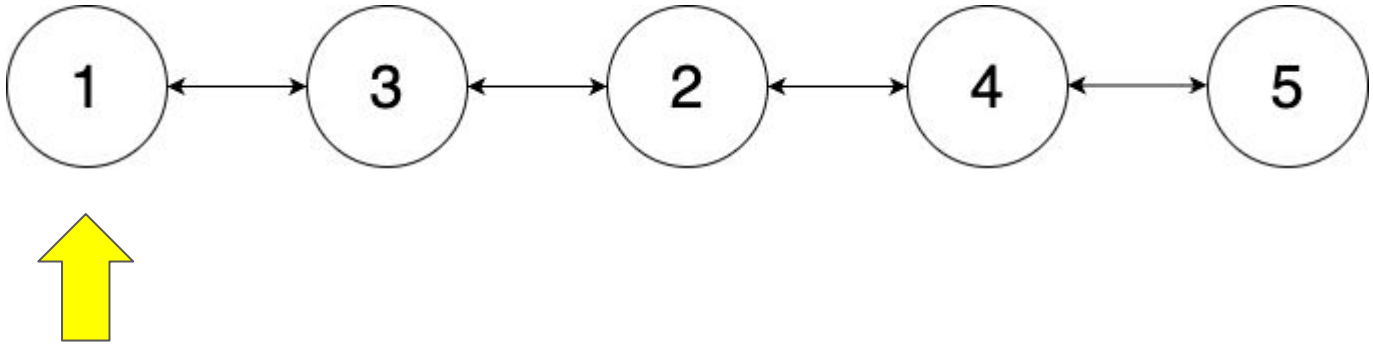
Newton School of Technology

Insertion node at kth postition in doubly linked list:



Insert value 7 at Position 3

Insertion node at kth postition in doubly linked list:

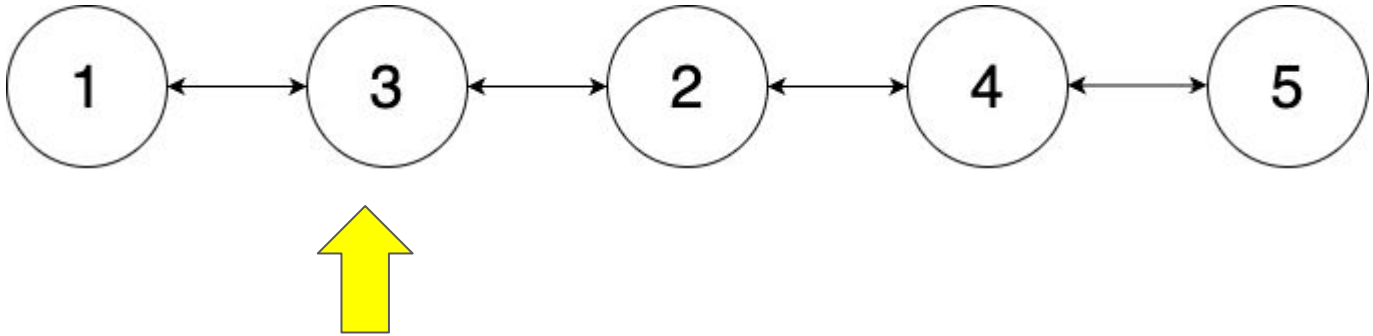


Counter is 1

Move to next node

Newton School of Technology

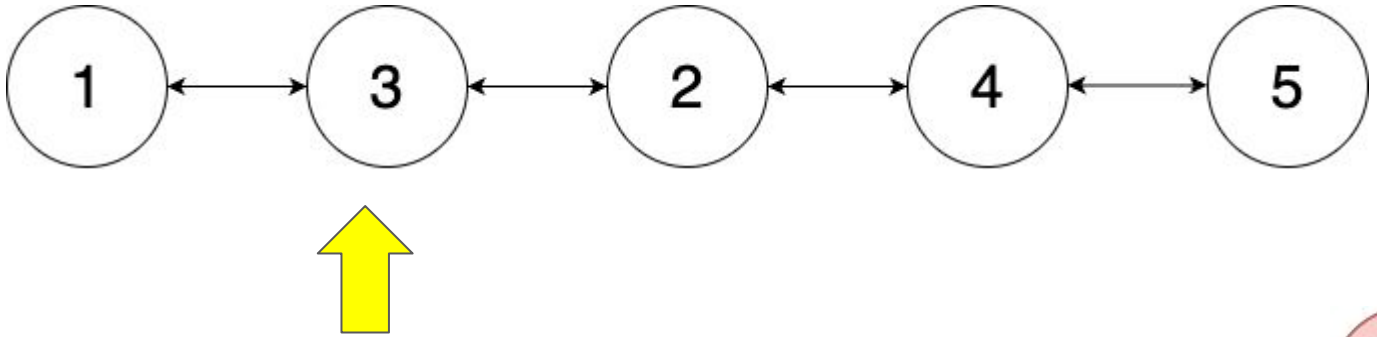
Insertion node at kth postition in doubly linked list:



Counter is 2

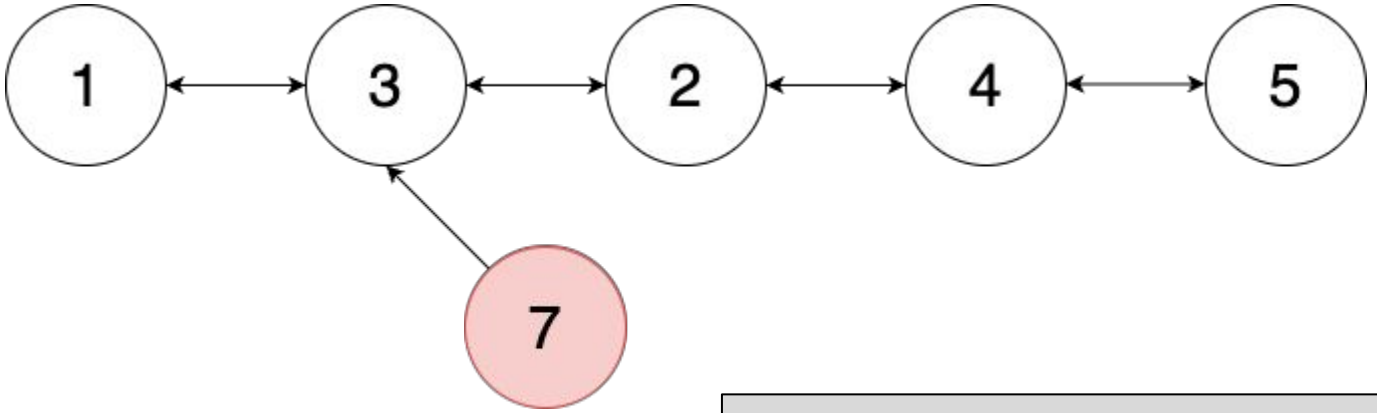
Stop here

Insertion node at kth postition in doubly linked list:



Create new_node with given value

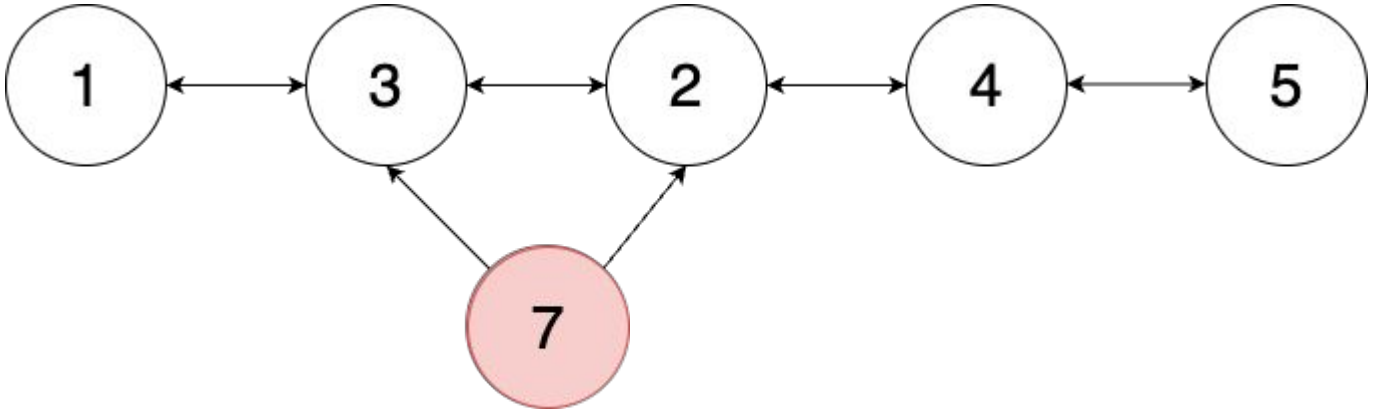
Insertion node at kth postition in doubly linked list:



Assign new_node.prev to cur node

Newton School of Technology

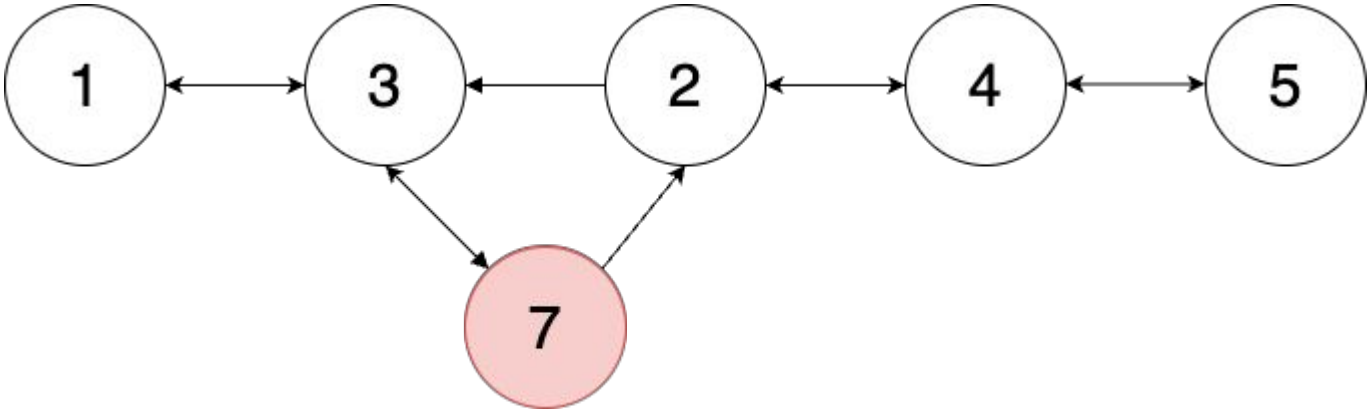
Insertion node at kth postition in doubly linked list:



Assign new_node.next to cur.next node

Newton School of Technology

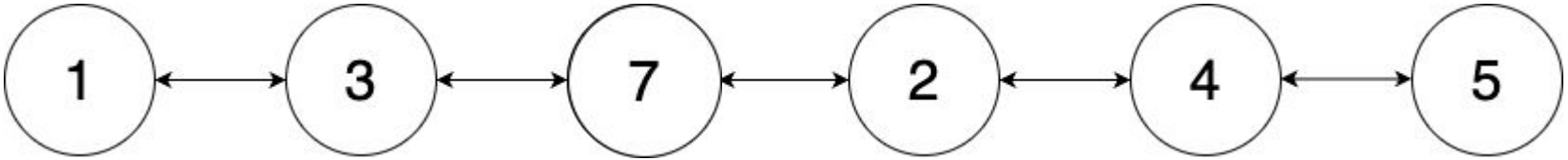
Insertion node at kth postition in doubly linked list:



Assign cur.next to new_node

Newton School of Technology

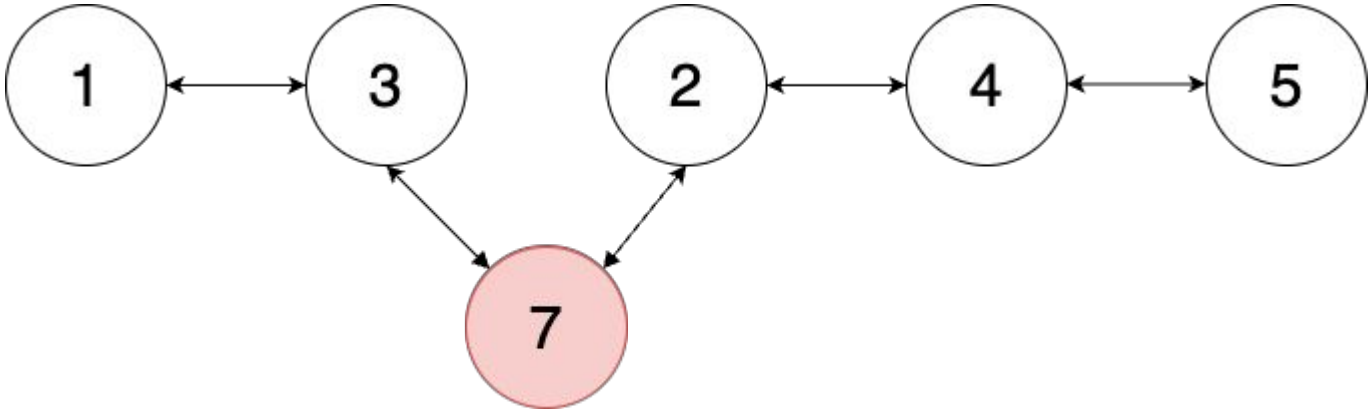
Insertion node at kth postition in doubly linked list:



A new_node with value 7 is inserted at pos. 3

Newton School of Technology

Insertion node at kth postition in doubly linked list:



Assign new_node.next.prev to itself

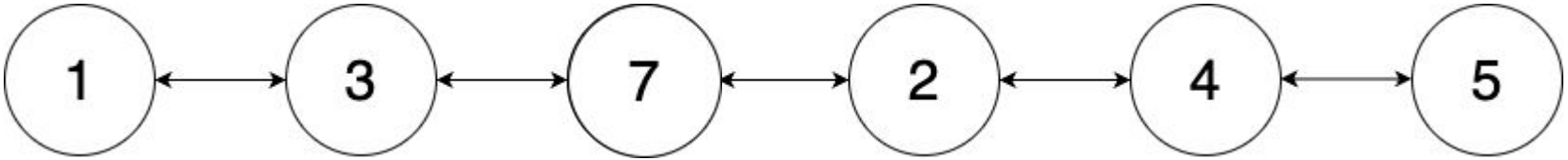
Newton School of Technology

Code :

```
def insertnew(head, k, pos):

    current = head
    cnt = 1
    while cnt != pos - 1 and current:
        current = current.next
        cnt += 1

    new_node = Node(k)
    new_node.next = current.next
    current.next.prev = new_node
    current.next = new_node
    new_node.prev = current

    return head
```

Newton School of Technology

Insertion node at kth postition in doubly linked list:



1. What will happen when pos=1
2. What will happen when pos=7

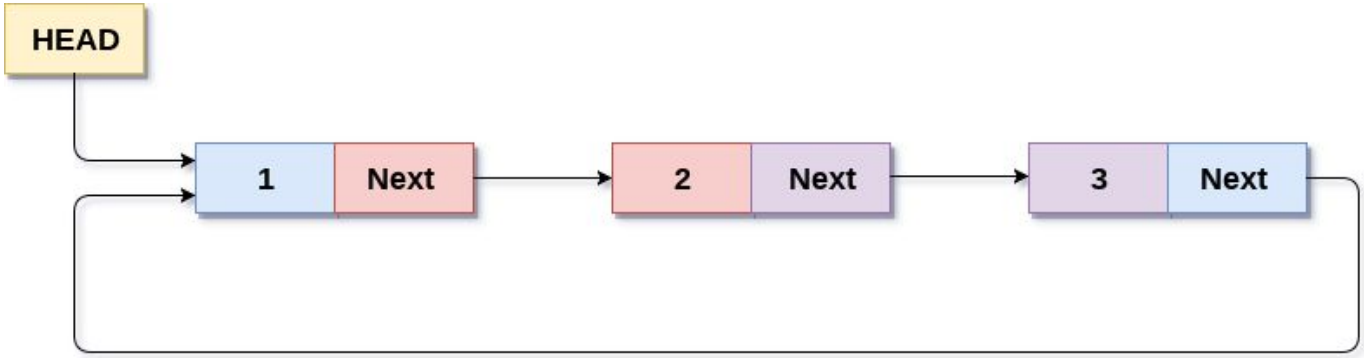
Newton School of Technology

Circular Linked List

Newton School of Technology

Circular Linked List :

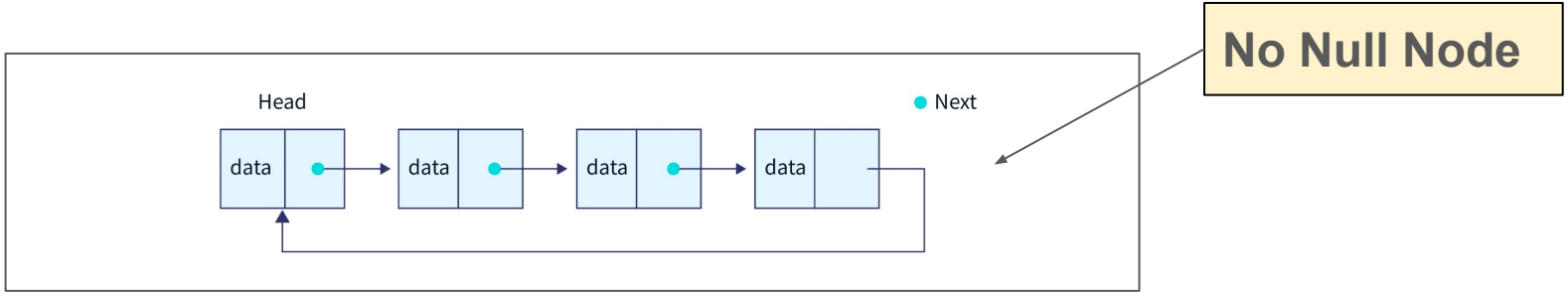
A Circular Linked List (CLL) is a linked list where the last node points back to the first node, forming a continuous loop without a NULL end.



Newton School of Technology

Node Structure of CLL :

```
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None  # Points to next node
```



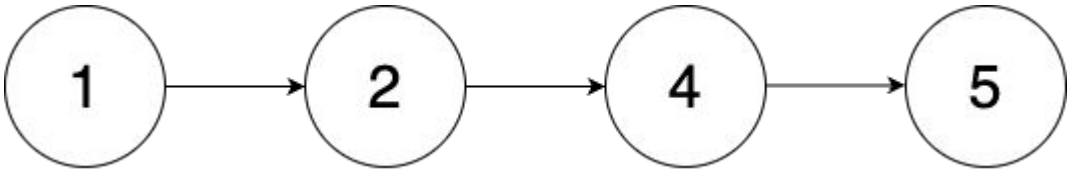
Newton School of Technology

Convert Singly LL to Circular LL

Newton School of Technology

Problem Statement :

Given a singly linked list, your task is to convert it into a circular linked list.



Newton School of Technology

How will you solve the problem ?

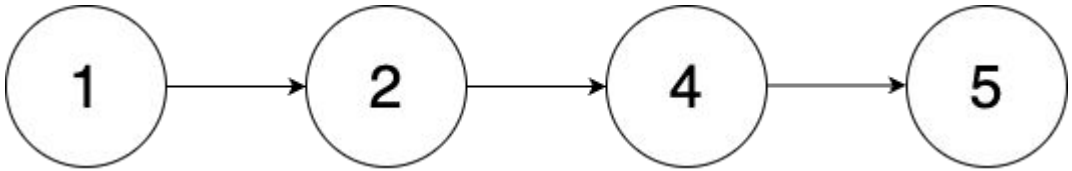
Newton School of Technology

Intuition

1. **Find Last Node:** Iterate till the last node
2. **Initialize Pointers:**
    - *initialize **last** node **next** pointer to **head** node*
3. ***Return** the **head** node*

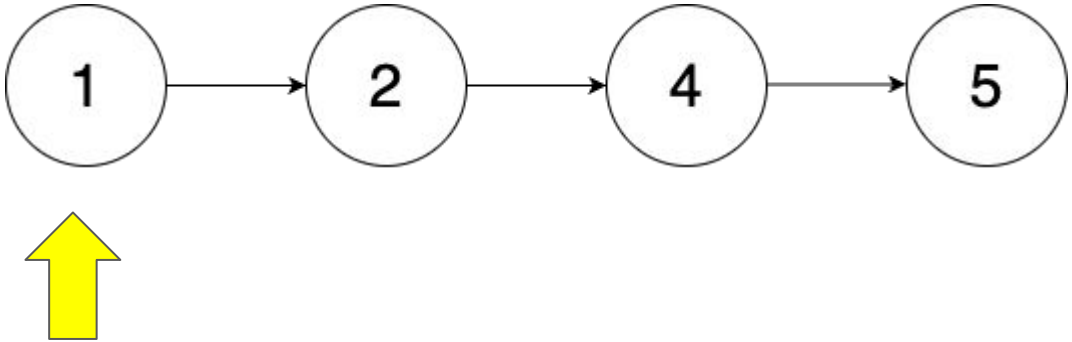
Newton School of Technology

Make it Circular:



Newton School of Technology

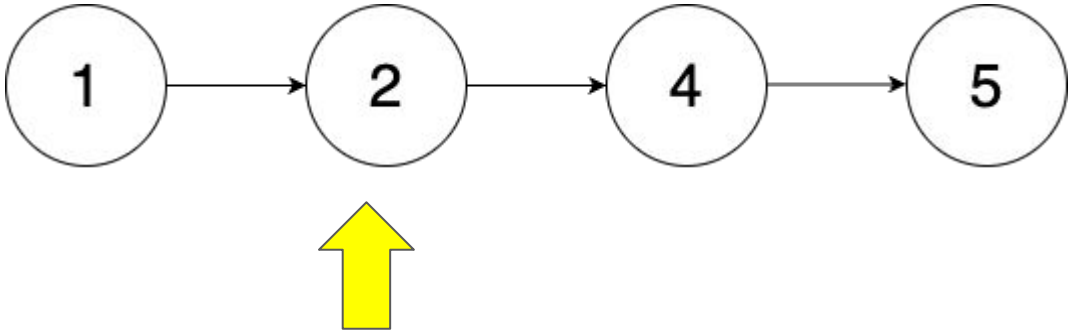
Make it Circular:



Node next is not Null

Jump to next node

Make it Circular:

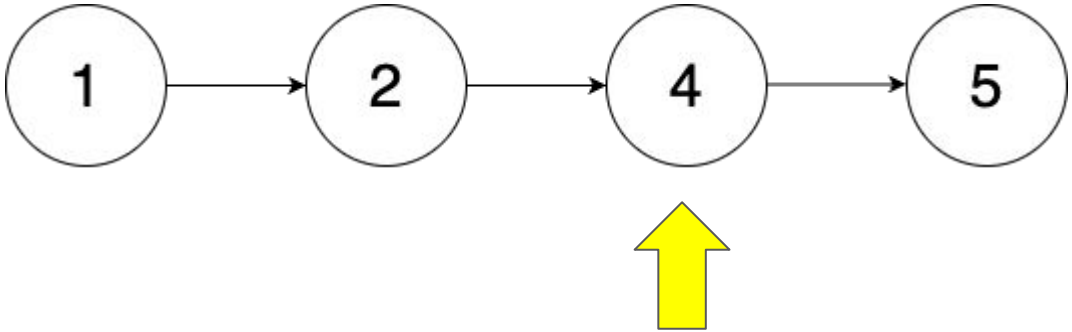


Node next is not Null

Jump to next node

Newton School of Technology

Make it Circular:

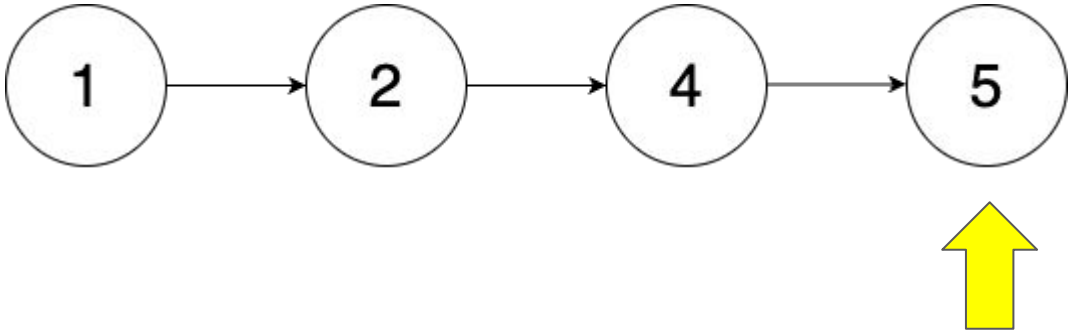


Node next is not Null

Jump to next node

Newton School of Technology

Make it Circular:

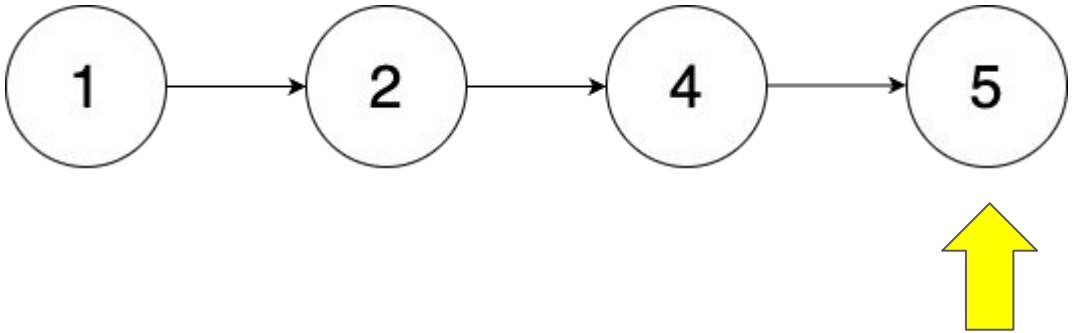


Node next is NULL

(Last Node)

Newton School of Technology

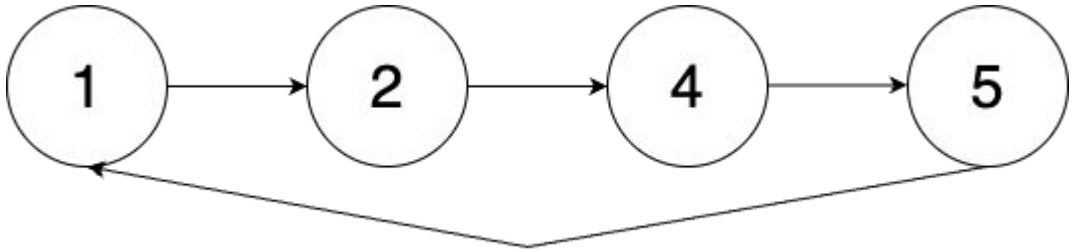
Make it Circular:



Assign it’s next pointer to HEAD node

Newton School of Technology

Make it Circular:



Assign it’s next pointer to HEAD node

Newton School of Technology

Code :

```
def make_circular(head):
    temp = head
    while temp.next is not None:
        temp = temp.next
    temp.next = head
    return head
```

Newton School of Technology

END