BEST Linked List Questions
Java

1. **Find the nth node from the end & remove it.**

Time complexity – O(n)

Space complexity – O(1)

```java
public ListNode removeNthFromEnd(ListNode head, int n) {
    if(head.next == null) {
        return null;
    }


    int size = 0;
    ListNode temp = head;
    while(temp != null) {
        temp = temp.next;
        size++;
    }

    //removing SIZEth node from last i.e. head
    if(n == size) {
        return head.next;
    }

    //find previous node
    int ptf = size - n; // position to find
    ListNode prev = head; // previous node
    int cp = 1; // current position

    while(cp != ptf) {
        prev = prev.next;
        cp++;
    }

    prev.next = prev.next.next;
    return head;
```

```
}
```

2. **Check if a Linked List is a palindrome**

Time complexity - O(n)

Space complexity - O(1)

```
public ListNode getMiddle(ListNode head) {
    ListNode fast = head;
    ListNode slow = head;
    while (fast.next != null && fast.next.next != null) {
        fast = fast.next.next;
        slow = slow.next;
    }
    return slow;
}

public ListNode reverse(ListNode head) {
    ListNode prev = null;
    ListNode curr = head;

    while (curr != null) {
        ListNode next = curr.next;
        curr.next = prev;
        prev = curr;
```

```
        curr = next;
    }
    return prev;
}



public boolean isPalindrome(ListNode head) {
    if(head == null || head.next == null) {
        return true;
    }

    ListNode firstHalfEnd = getMiddle(head);
    ListNode secondHalfStart = reverse(firstHalfEnd.next);
    ListNode firstHalfStart = head;

    while(secondHalfStart != null) {
        if(secondHalfStart.val != firstHalfStart.val) {
            return false;
        }
        secondHalfStart = secondHalfStart.next;
        firstHalfStart = firstHalfStart.next;
    }

    return true;
}
```

3. **Detecting Loop in a Linked List.**

Time complexity - O(n)

Space complexity - O(1)

```java
public boolean hasCycle(ListNode head) {

   ListNode slow = head;

   ListNode fast = head;



   while(fast != null && fast.next != null) {

       slow = slow.next;

       fast = fast.next.next;



       if(fast == slow) {

           return true;

       }

   }



   return false;

}
```

Homework Problems

1. Removing Loops in a Linked List.

   (Please try on your own first. The answer will be updated soon!)