```
class LinkedListNode {
    constructor (val) {
        this.value = val
        this.next = null
    }
}

function removeNthFromLinkedList(head, n) {
    if (!head || n <= 0) return head;

    if (n === 1) {
        head = head.next;
        return head;
    }

    let current = head;
    let prev = null;
    let index = 1;

    while (current && index < n) {
        prev = current;
        current = current.next;
        index++;
    }

    if (!current) return head; // If n is out of bounds

    prev.next = current.next;

    return head;
}

function insertIntoLinkedList (head, t) {

    // make the value a proper linked list node
    const nodeToInsert = new LinkedListNode(t)

    // if head does not exist or t < head, make t head
    if (!head || nodeToInsert.value < head.value) {
        nodeToInsert.next = head
        return nodeToInsert
    }

    let current = head

    while (current.next && nodeToInsert.value > current.next.value) {
        current = current.next // way to iterate
    }

    nodeToInsert.next = current.next
    current.next = nodeToInsert
```

```
    return head
}

let head = null

head = insertIntoLinkedList(head, 2)
head = insertIntoLinkedList(head, 3)
head = insertIntoLinkedList(head, 4)

let current = head;
let result = "";
while (current) {
    result += current.value + " ";
    current = current.next;
}
console.log(result);

head = removeNthFromLinkedList (head, 1)

current = head
result = "";
while (current) {
    result += current.value + " ";
    current = current.next;
}
console.log(result);
```