Logic:

1. If val < head then place it at the beginning and make it the head node
2. Run through the values in the list starting from the head node and find the suitable node to add it after
3. After finding the correct node, insert the input node.

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

function insertIntoLinkedList (head, t) {

    // make the value a proper linked list node
    const nodeToInsert = new Node(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);
```

If completely class based, then only change is:

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

class LinkedList {
    constructor() {
        this.head = null; //**** This line will be added
    }

    insertSorted(data) {

        let newNode = new Node(data);
        /...

        if (!this.head || data < this.head.value) {
            newNode.next = this.head;
            this.head = newNode; //**** This line will be added
            return;
        }

        /...
    }

    printList() {}
}

let list = new LinkedList();
list.insertSorted(5);
list.insertSorted(3);
```