Reverse a LinkedList using recursion , given is headNode

```
class LinkedListNode {
    // let arr = [];
    constructor(value, next){
        this.value = value;
        this.next = (next == null || next == undefined) ? null: next;
    }


    func(x,y){
        return x+y;
    }
}

let headNode = new LinkedListNode(10);
headNode.next = new LinkedListNode(20);
headNode.next.next = new LinkedListNode(30);

// console.log(headNode);


// const reverseFunc = (headNode) => {

//     if(headNode == null || headNode.next == null)
//         return headNode;

//     let newHead = reverseFunc(headNode.next);
//     headNode.next.next = headNode;
//     headNode.next = null;

//     return newHead;
// }

// console.log("");

// console.log("Reversed LinkedList");
// console.log(reverseFunc(headNode));


// Add 2 non- negative linkedList where digits are stored in reverse order, and each of their nodes contains a single digit. Add the two numbers and return the sum as a linked list.
```

```
let head1 = new LinkedListNode(1);
head1.next = new LinkedListNode(2);
head1.next.next = new LinkedListNode(8);


let head2 = new LinkedListNode(7);
head2.next = new LinkedListNode(8);
head2.next.next = new LinkedListNode(4);


const addLinkedList = (head1, head2) => {

  const recursiveFunc = (head1, head2, carry) => {
    let currSum = (head1 && head1.value || 0 )  +  (head2 && head2.value || 0 ) + carry;

    let currCarry = Math.floor(currSum/10);
    let currNodeValue = currSum%10;

    return (head1 || head2 || carry) ?
     new LinkedListNode(currNodeValue, recursiveFunc(head1 && head1.next || null , head2
&& head2.next || null, currCarry))  : null;
  }


  return recursiveFunc(head1, head2, 0);
}

let newLLHead = addLinkedList(head1, head2);


while(newLLHead != null){
  console.log(newLLHead.value);
  newLLHead = newLLHead.next;
}
```