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

class LinkedList {
  constructor(data) {
    this.head = null;
  }

  addFirst(data) {
    const newNode = new Node(data);
    newNode.next = this.head;
    this.head = newNode;
  }

  size() {
    let count = 0;
    if (!this.head) {
      return count;
    }
    let current = this.head;

    while (current) {
      count++;
      current = current.next;
    }
    return count;
  }

  print() {
    let current = this.head;
    while (current) {
      console.log(current.data);
      current = current.next;
    }
  }
}

class SortLinkedList extends LinkedList {
  sort() {
    this.head = this.mergeSort(this.head);
  }

  mergeSort(node) {
    if (!node || !node.next) {
      return node;
    }

    let middle = this.getMiddle(node);
    let nextOfMiddle = middle.next;
```

```
    middle.next = null;

    let left = this.mergeSort(node);
    let right = this.mergeSort(nextOfMiddle);

    let sortedList = this.sortedMerge(left, right);
    return sortedList;
  }

  getMiddle(node) {
    if (!node) {
      return node;
    }
    let slow = node;
    let fast = node.next;

    while (!fast && fast.next !== null) {
      slow = slow.next;
      fast = fast.next.next;
    }
    return slow;
  }
  sortedMerge(a, b) {
    let result = null;
    if (!a) {
      return b;
    }
    if (!b) {
      return a;
    }

    if (a.data <= b.data) {
      result = a;
      result.next = this.sortedMerge(a.next, b);
    } else {
      result = b;
      result.next = this.sortedMerge(a, b.next);
    }

    return result;
  }
}
class LinkedListWithPrint extends LinkedList {
  setHead(node) {
    this.head = node;
  }
}
const sortedLinkedList = new SortLinkedList();

sortedLinkedList.addFirst(3);
sortedLinkedList.addFirst(13);
sortedLinkedList.addFirst(-8);
sortedLinkedList.addFirst(5);
sortedLinkedList.sort();
```

```
const sortedLinkedListTwo = new SortLinkedList();
sortedLinkedListTwo.addFirst(-30);
sortedLinkedListTwo.addFirst(123);
sortedLinkedListTwo.addFirst(88);
sortedLinkedListTwo.addFirst(50);
sortedLinkedListTwo.sort();

const sortedLinkedListThree = new SortLinkedList();

// function mergeTwoSorted(list1, list2) {
//   let result = null;
//   if (!list1) {
//     return list2;
//   }
//   if (!list2) {
//     return list1;
//   }
//   if (list1.data < list2.data) {
//     result = list1;
//     result.next = mergeTwoSorted(list1.next, list2);
//   } else {
//     result = list2;
//     result.next = mergeTwoSorted(list2.next, list1);
//   }
//   return result;
// }

const mergedListHead = sortedLinkedListThree.sortedMerge(
  sortedLinkedList.head,
  sortedLinkedListTwo.head
);
//console.log("mergedListHead",mergedListHead)
const mergedLinkedList = new LinkedListWithPrint();
mergedLinkedList.setHead(mergedListHead);
mergedLinkedList.print();
```