FLATTENING A LINKED LIST

☆ In this problem we are given a 2D Linked list where each node has a vertical extension as well. Our job is to return a vertically flatenned linked list which is sorted.

Bruteforce Solution is to flatten the entire linked list into an array, sort the array, convert it to vertical linked list and return it.

Optimal Solution involves the use of property of linked list being vertically sorted and do all changes in place. Merging two vertical parts is very similar to merging two sorted LLs.

```
Node * merge2Vertically(Node * l1, Node * l2){
    Node * dN = new Node(-1);
    Node * t = dN;
    while(l1 != nullptr && l2 != nullptr){
        if(l1 -> val < l2 -> val) {
            t -> child = l1;
            t = l1;
            l1 = l1 -> child;
        } else {
            t -> child = l2;
            t = l2;
```

```
                l2 = l2 -> child ;
            }
            t -> next = nullptr ;
        }
        if (l1 == nullptr) {
            t -> child = l2 ;
        }
        if (l2 == nullptr) {
            t -> child = l1 ;
        }
    return dN -> child ;
}
```

We can use this function to flatten multiple vertical parts into 1 by taking 2 in pairs.

```
Node * flattenOutLL(Node * head) {
    if (head == nullptr || head -> next
                                == nullptr) {
        return head ;
    }
    Node * m = flattenOutLL(head -> next);
    return merge2Vertically(head , m) ;
}
```