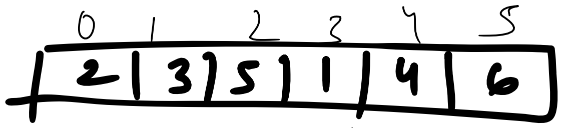
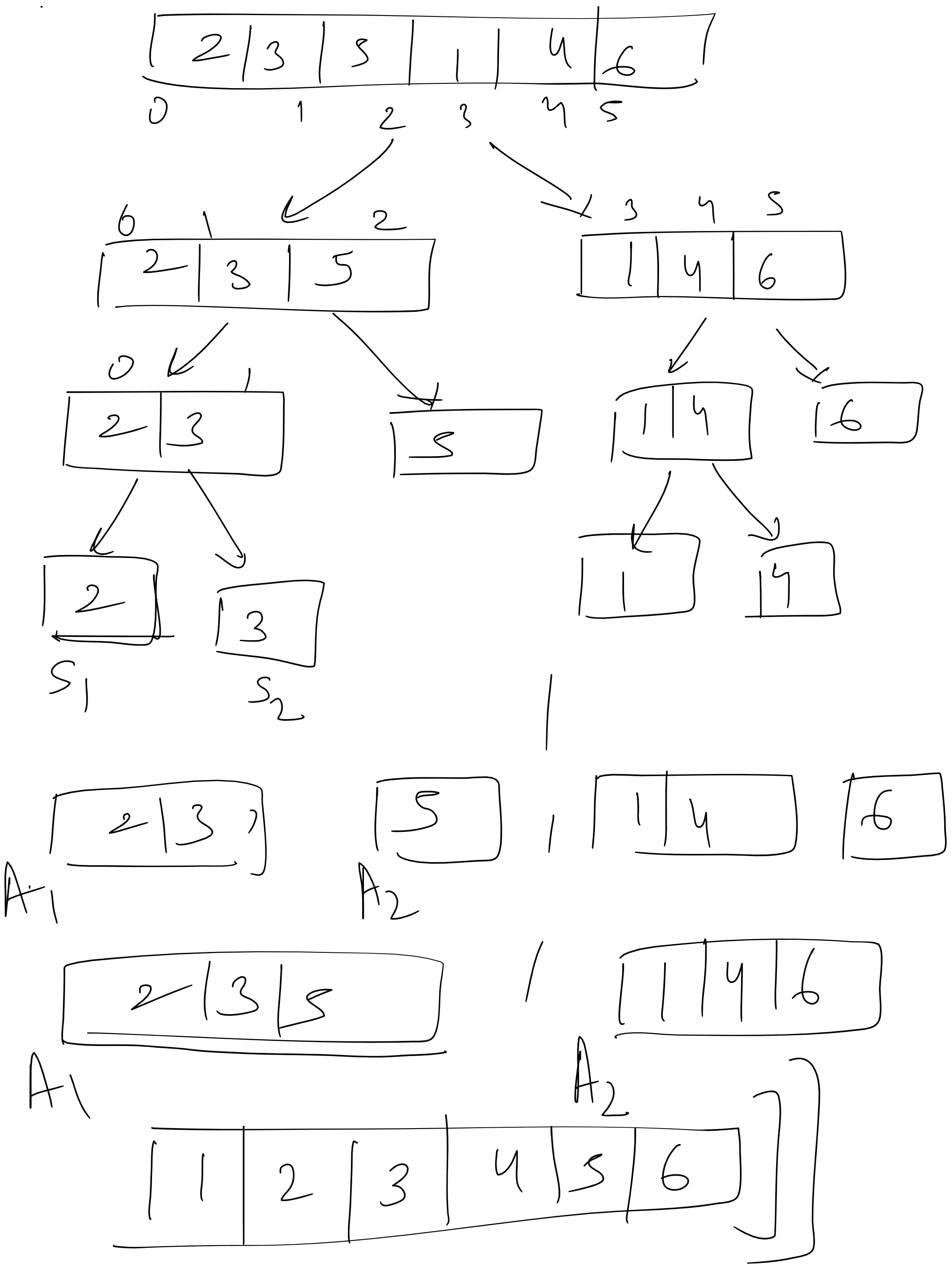
Day 18

06 December 2022 16:01

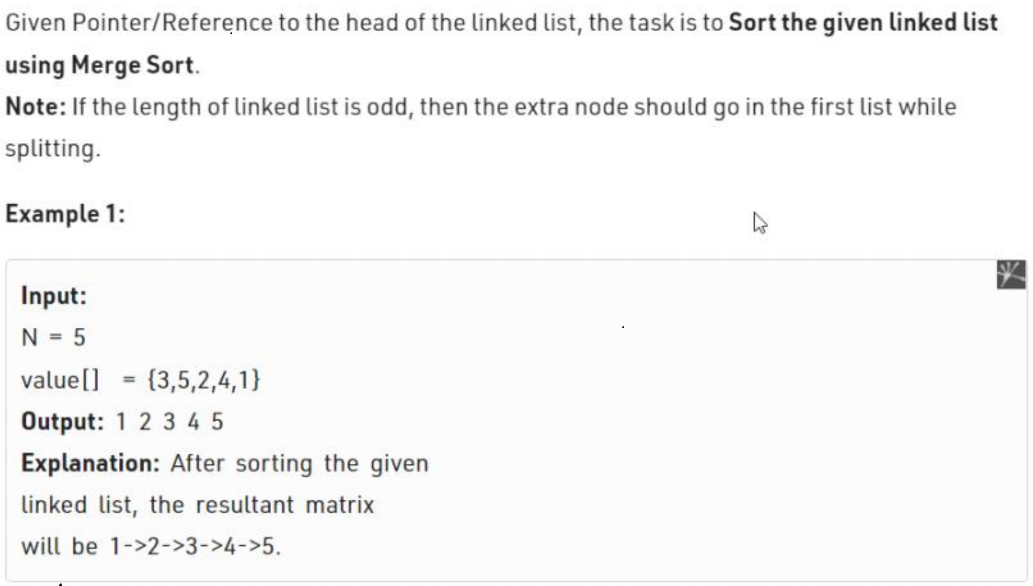


Divide and Conquer



T.C - $O(n \log n)$

S. C – O(N)



Given Pointer/Reference to the head of the linked list, the task is to **Sort the given linked list using Merge Sort**.

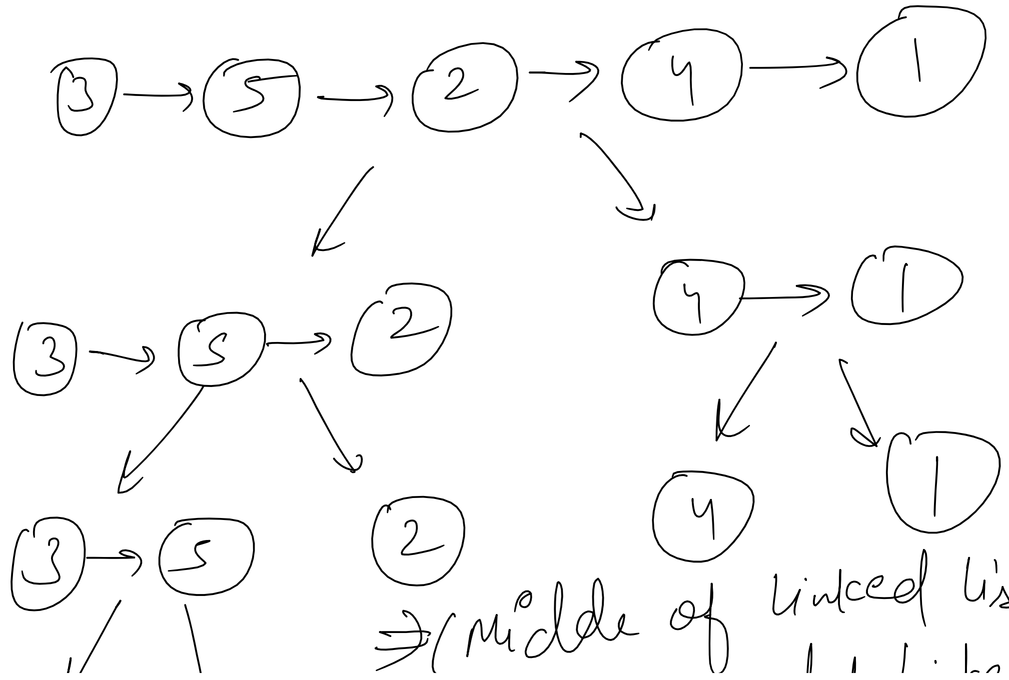
Note: If the length of linked list is odd, then the extra node should go in the first list while splitting.

Example 1:

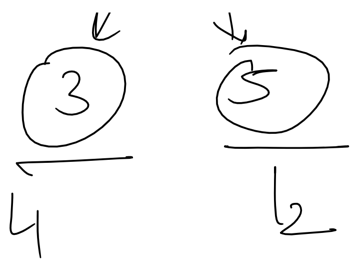
Input:
N = 5
value[] = {3,5,2,4,1}
Output: 1 2 3 4 5
Explanation: After sorting the given linked list, the resultant matrix will be 1->2->3->4->5.

I/P
$$\begin{tikzcd} 3 \arrow[r] & 5 \arrow[r] & 2 \arrow[r] & 4 \arrow[r] & 1 \end{tikzcd}$$

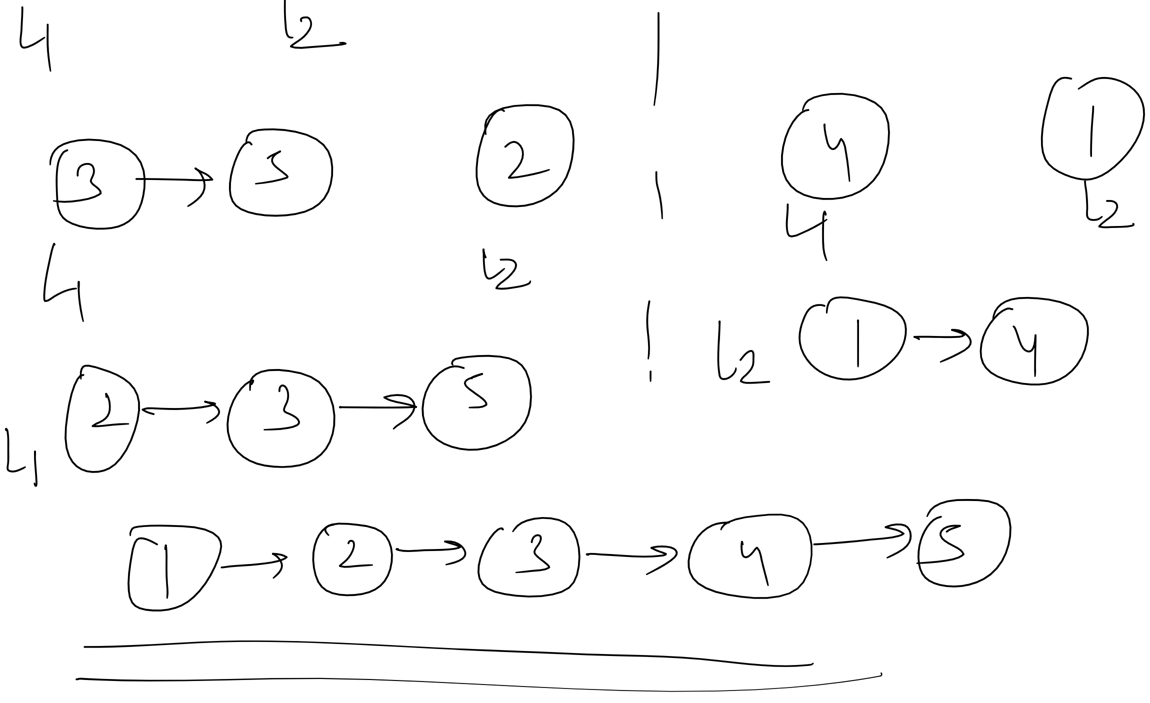
O/P
$$\begin{tikzcd} 1 \arrow[r] & 2 \arrow[r] & 3 \arrow[r] & 4 \arrow[r] & 5 \end{tikzcd}$$



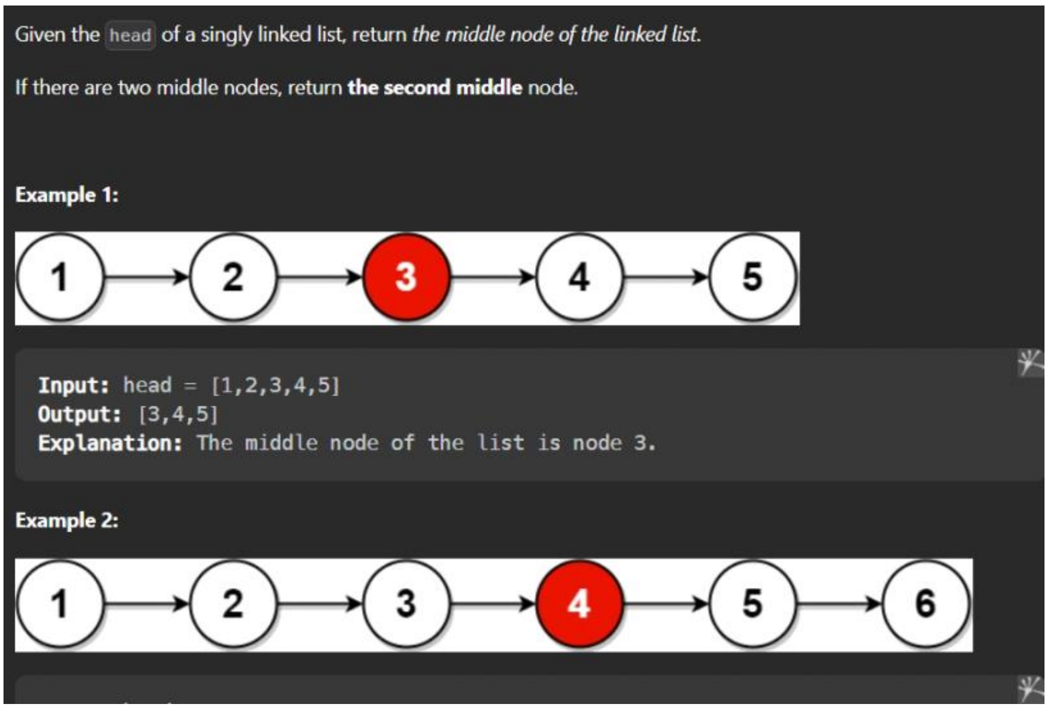
⇒(Middle of linked list)



→ (merge two sorted linked list)



Middle of linked list



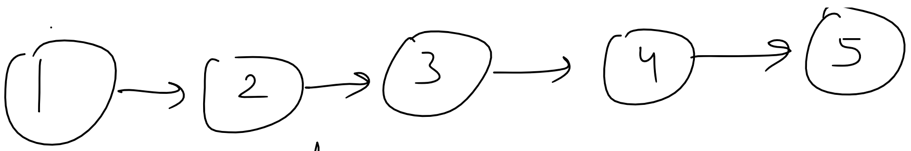
Given the `head` of a singly linked list, return *the middle node of the linked list*.

If there are two middle nodes, return **the second middle** node.

Example 1:

```
Input: head = [1,2,3,4,5]
Output: [3,4,5]
Explanation: The middle node of the list is node 3.
```

Example 2:

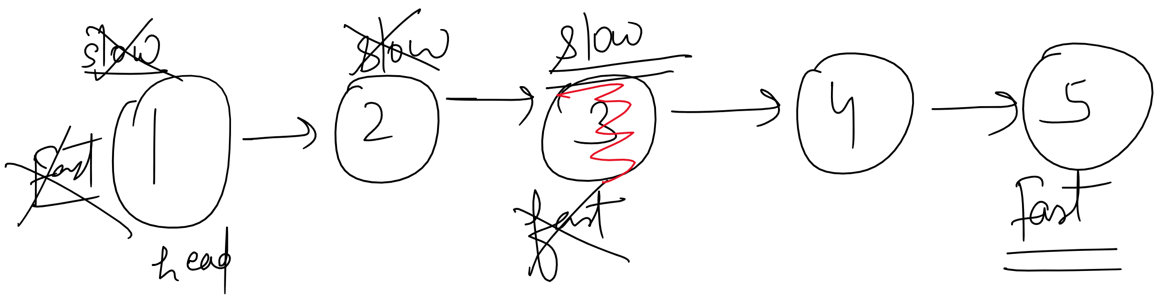


Initially

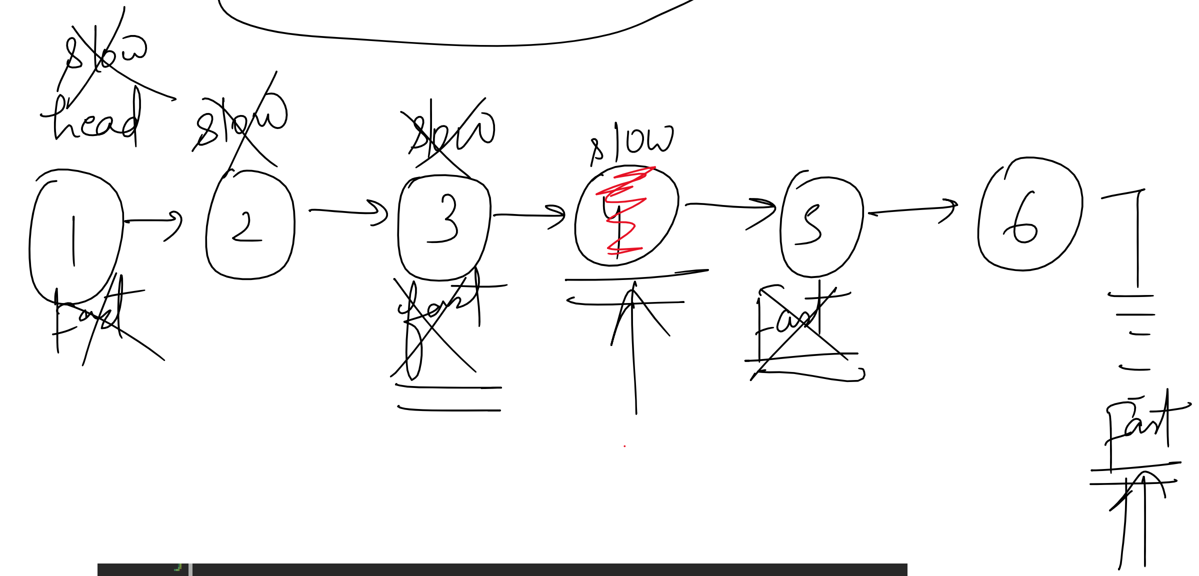
slow
fast } → head

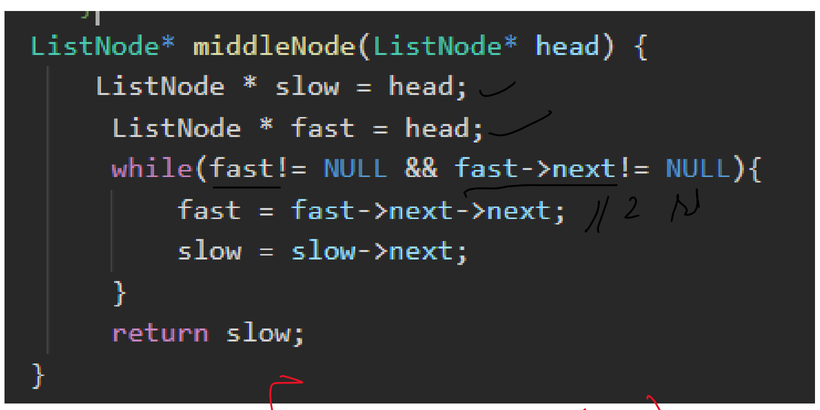
|| slow = 1 step
Fast = 2 step

Fast → last node
↓
NULL



T.c − $O(n/2)$

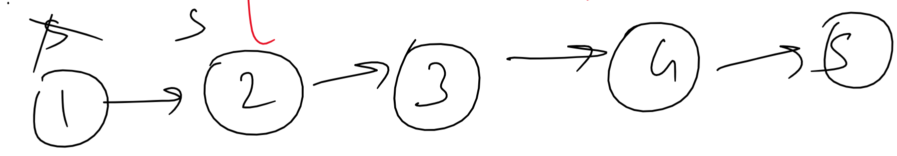




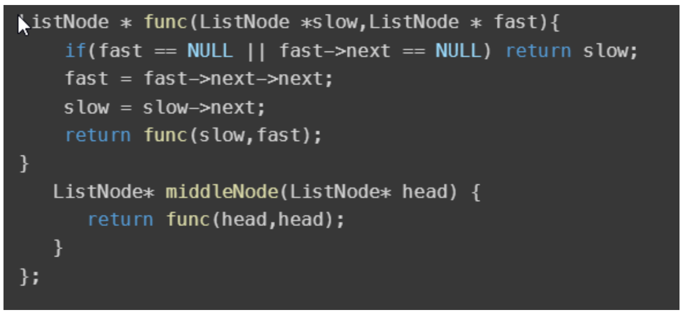
```
ListNode* middleNode(ListNode* head) {
    ListNode * slow = head;
     ListNode * fast = head;
     while(fast!= NULL && fast->next!= NULL){
         fast = fast->next->next; // 2 N
         slow = slow->next;
     }
     return slow;
}
```

//2 steps
// 1 step

[T.c − O(n) S.c − O(1)]



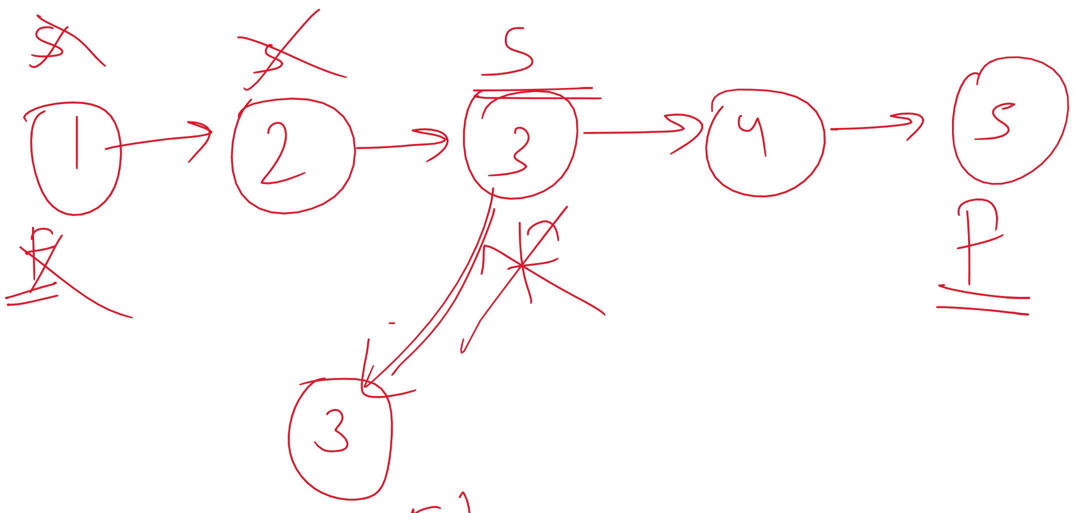
~~F~~ F



```
ListNode * func(ListNode *slow,ListNode * fast){
    if(fast == NULL || fast->next == NULL) return slow;
    fast = fast->next->next;
    slow = slow->next;
    return func(slow,fast);
}
   ListNode* middleNode(ListNode* head) {
      return func(head,head);
   }
};
```

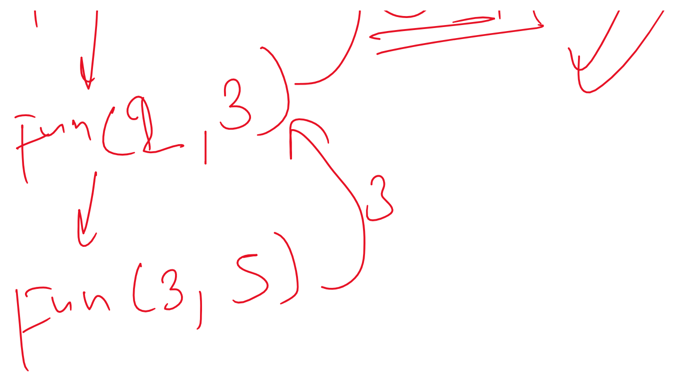
T.C — O(n)

S.C — O(n)



func(S, F)

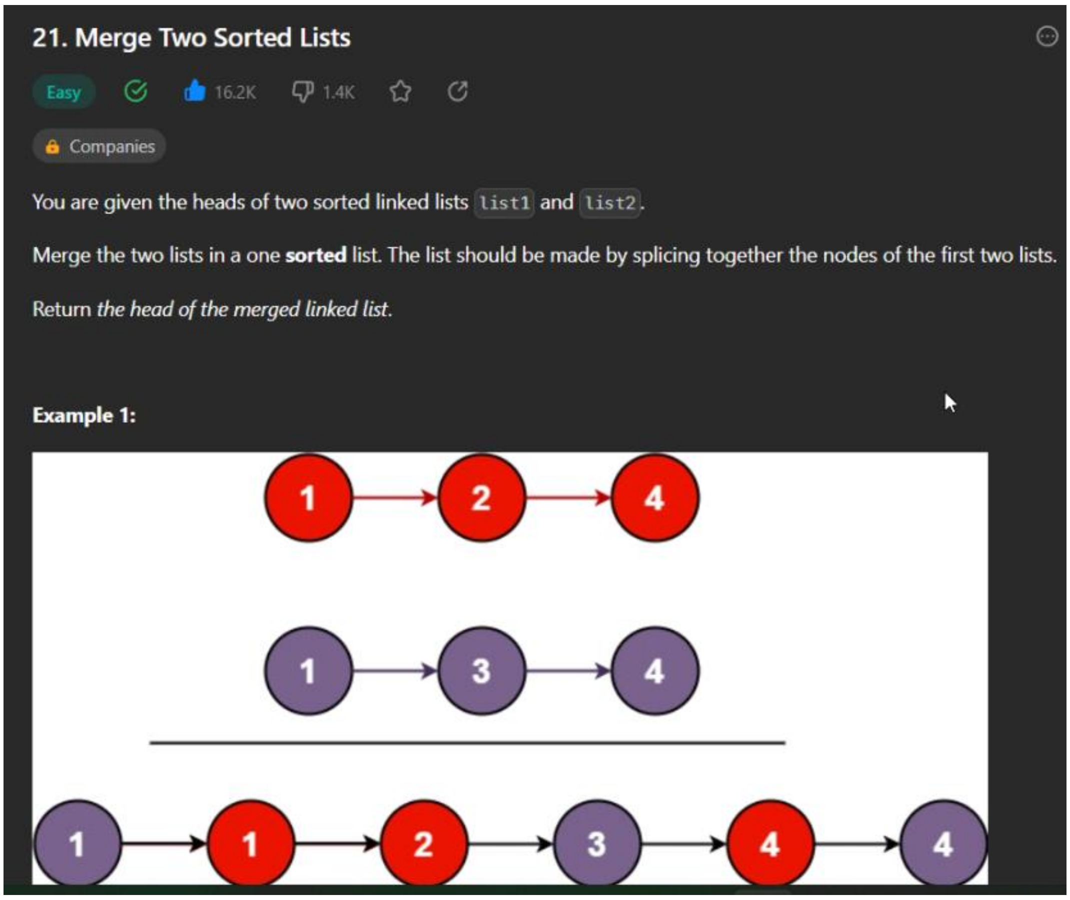
func(1,1)
3
func(2, 3)



// Loop के अंदर जो लिखा हुआ है वो recursive function के अंदर as it is likh do

// Loop terminating condition को Base case बना दो ||

// next set of input recursive call कर दो ||



21. Merge Two Sorted Lists

Easy 16.2K 1.4K

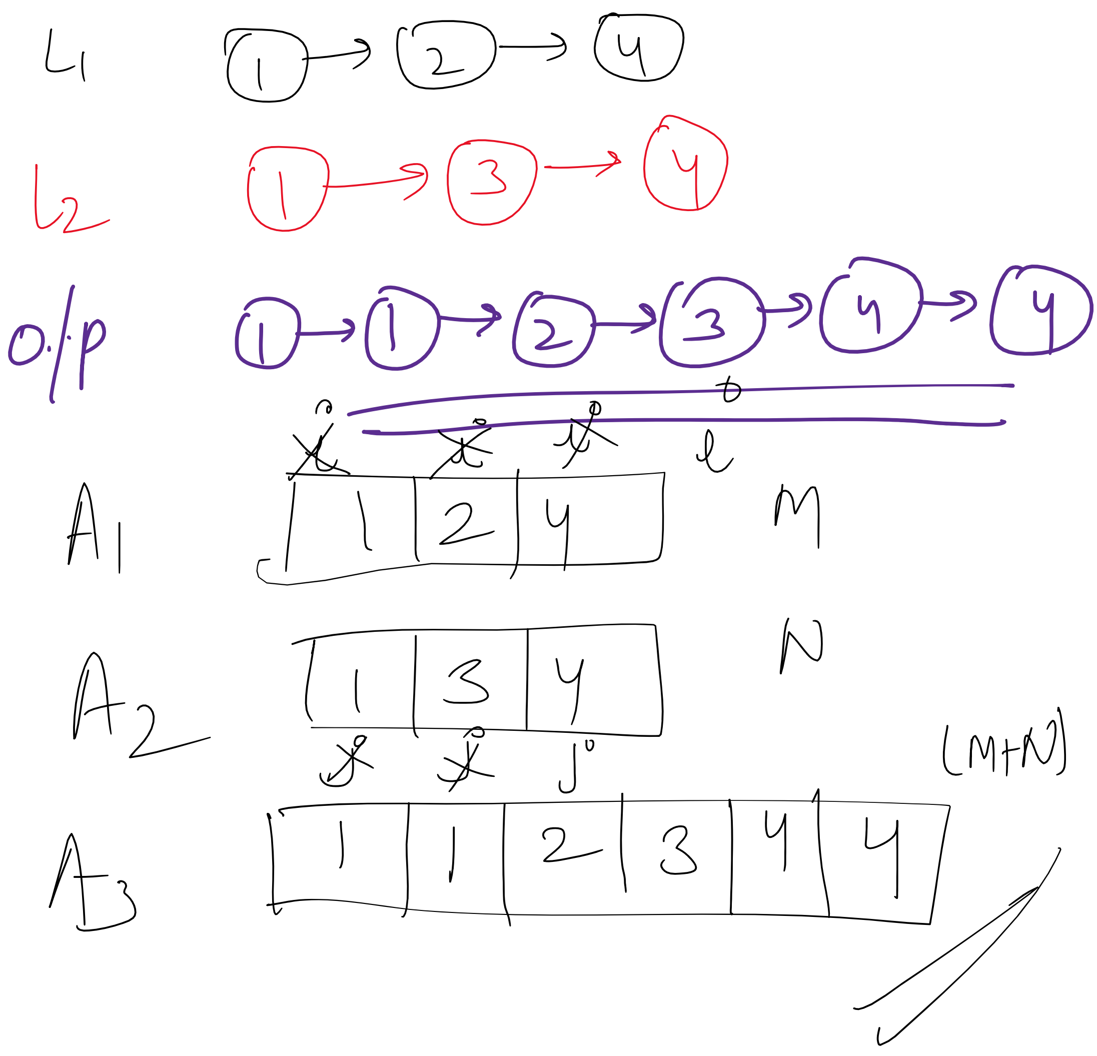
Companies

You are given the heads of two sorted linked lists `list1` and `list2`.

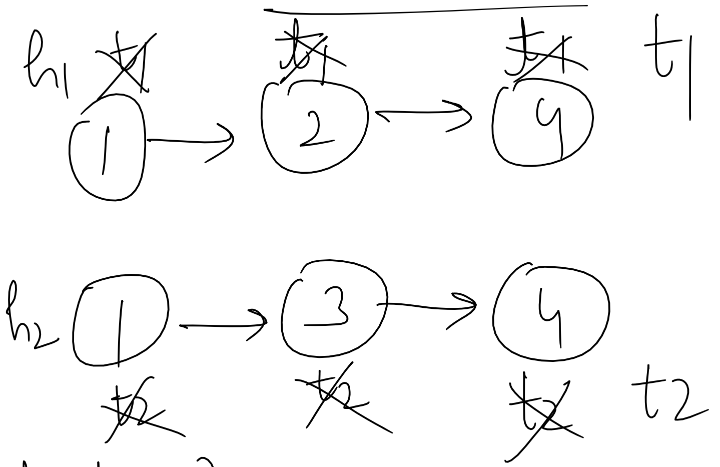
Merge the two lists in a one **sorted** list. The list should be made by splicing together the nodes of the first two lists.

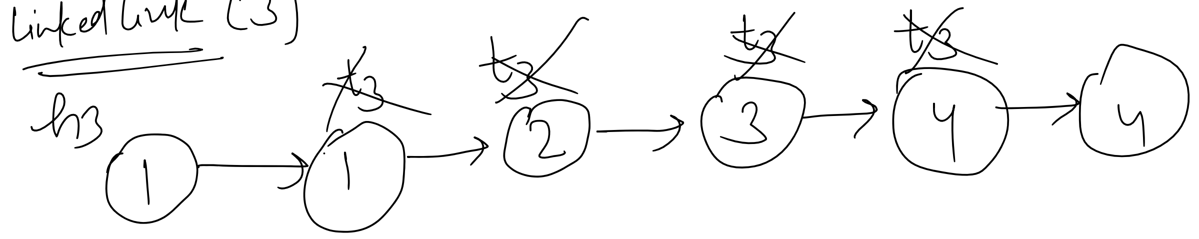
Return *the head of the merged linked list.*

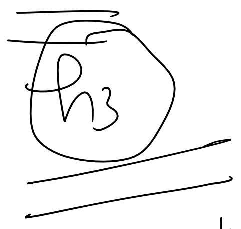
Example 1:

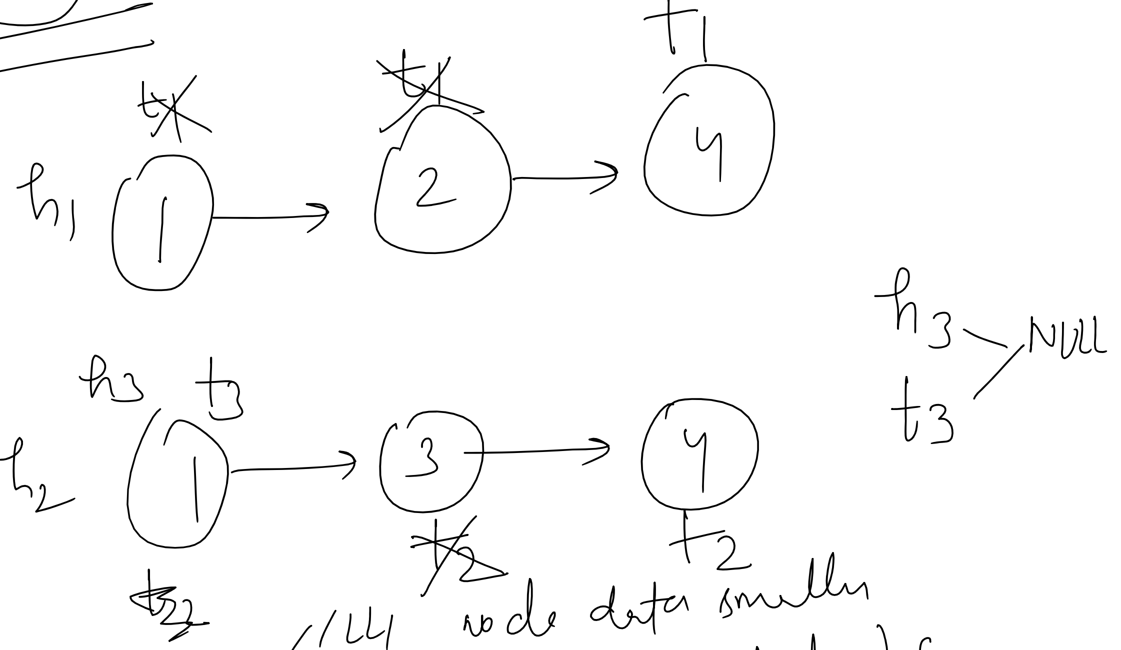


No extra space





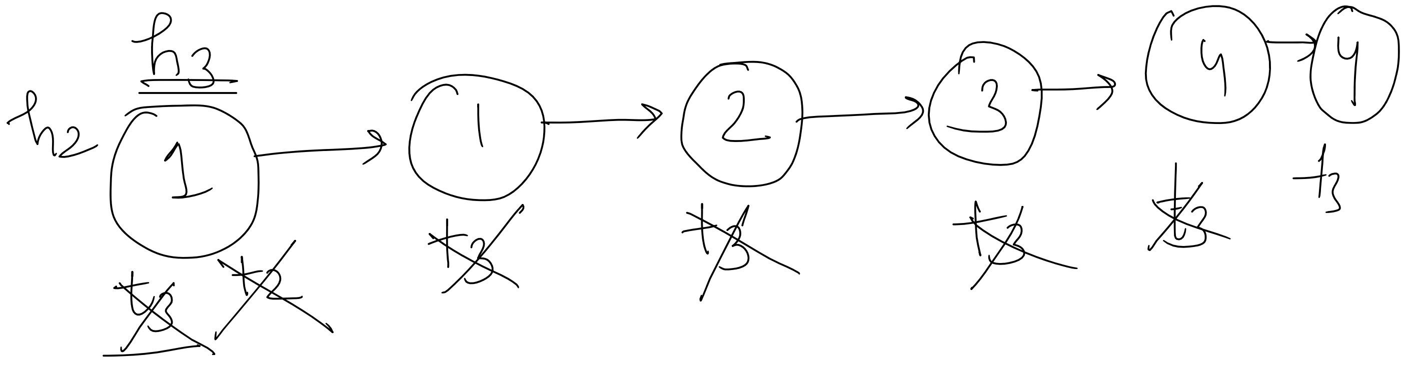


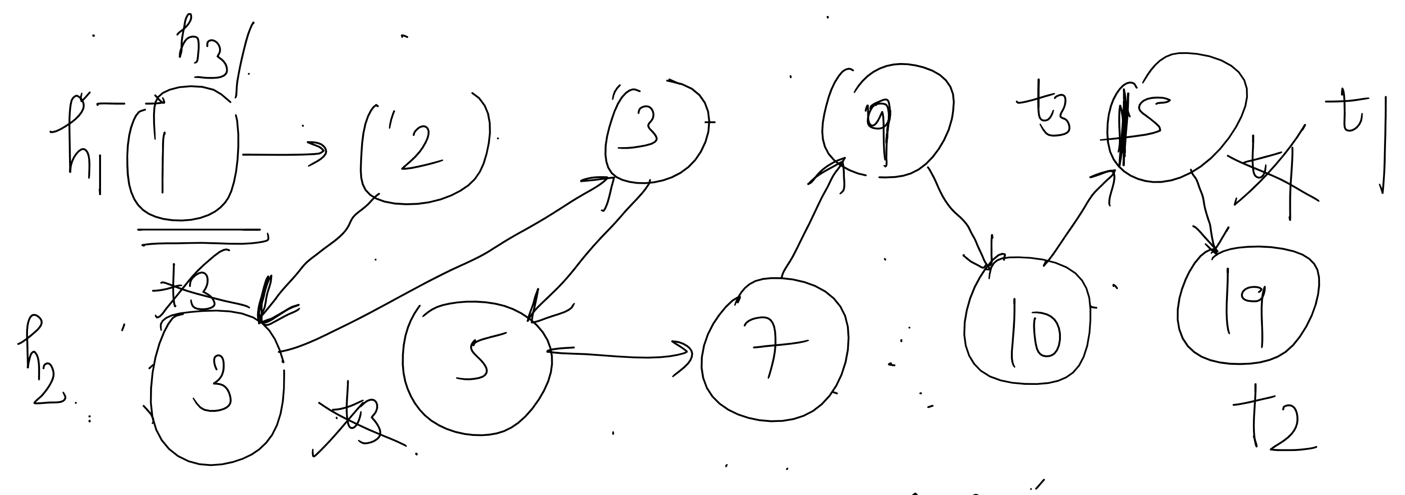


```
//LL1 node data smaller
if (t1->data < t2->data){
    if(h3 == NULL){ h3 = t3 = t1;}
    else{ t3->next = t1; t3 = t3->next;
    }
}else{ if(h3==NULL)
```

$\{ h_3 = t_3 = t_2 ;$

$\}$





Merge sorted linked list (h_3)

h_3 t_3

```
if (t1 == NULL)  t3 -> next = t2;
if (t2 == NULL)  t3 -> next = t1;
```

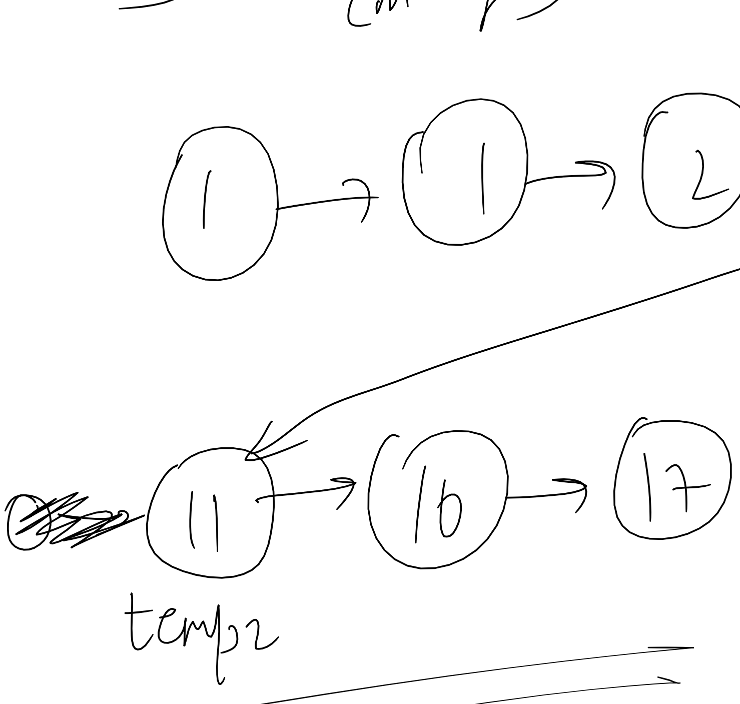
```
ListNode* temp1 = head1, *temp2 = head2;
ListNode* h3 = NULL, *t3 = NULL;
while( temp1 != NULL &&
        temp2 != NULL) {
  if (temp1 -> data < temp2 -> data){
        if (h3 == NULL) {
            h3 = t3 = temp1;
        }else {
            t3 -> next = temp1;
            t3 = t3 -> next;
        }
      temp1 = temp1 -> next;
  }
  else {

      if (h3 == NULL) {
          h3 = t3 = temp2;
      }else {
          t3 -> next = temp2;
          t3 = t3 -> next;
      }
      temp2 = temp2 -> next;
   }
}
```

}

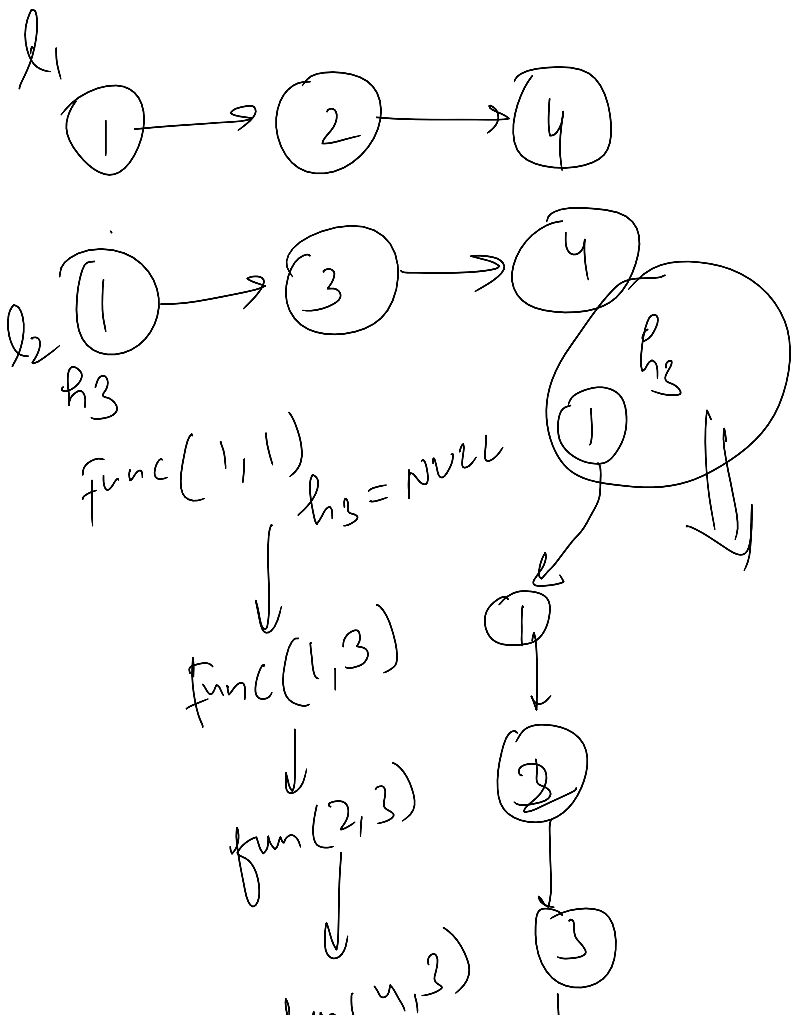
if (temp1 == NULL) t3 -> next = temp2;
if (temp2 == NULL) t3 -> next = temp1;

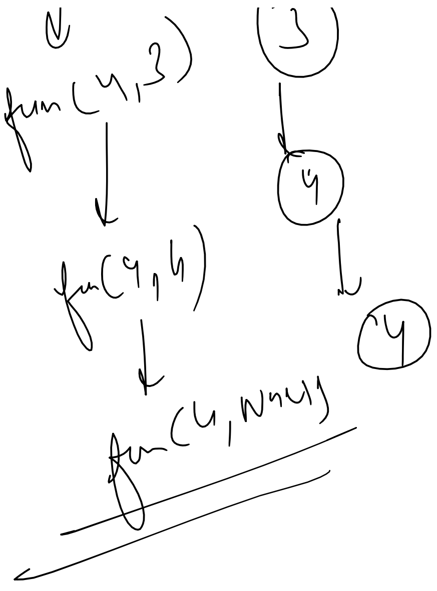
return h3;

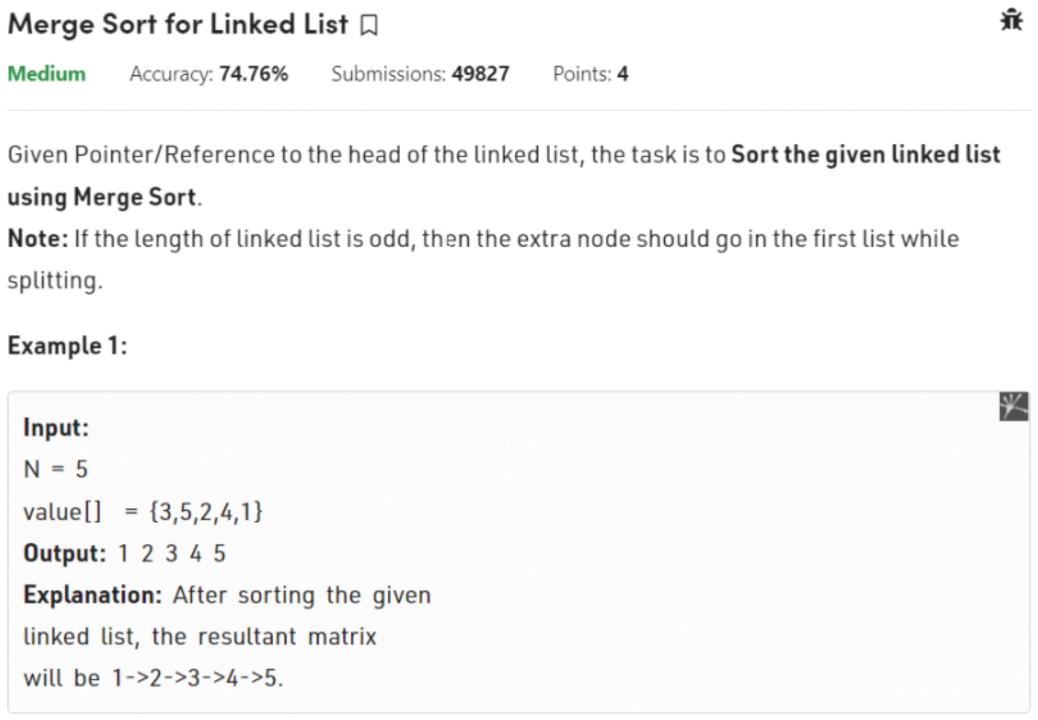
}   (merge)



```
 ListNode* mergeTwoLists(ListNode* list1, ListNode* list2)
{
        if(list1 == NULL) return list2;
        if(list2 == NULL) return list1;
        ListNode *temp1 = list1,*temp2= list2;
        ListNode *h3 = NULL,*t3 = NULL;
        while(temp1 != NULL && temp2 != NULL){
            if(temp1->val < temp2->val){
                if(h3 == NULL){
                    h3 = t3 = temp1;
                }else{
                    t3->next = temp1;
                    t3 = t3->next;
                }
                temp1 = temp1->next;
            }else{
                if(h3 == NULL){
                    h3 = t3 = temp2;
                }else{
                    t3->next = temp2;
                    t3 = t3->next;
                }
                temp2 = temp2->next;
            }
        }
        if(temp1 == NULL){
            t3->next = temp2;
        }else{
             t3->next = temp1;
        }
        return h3;
    }
```







Merge Sort for Linked List

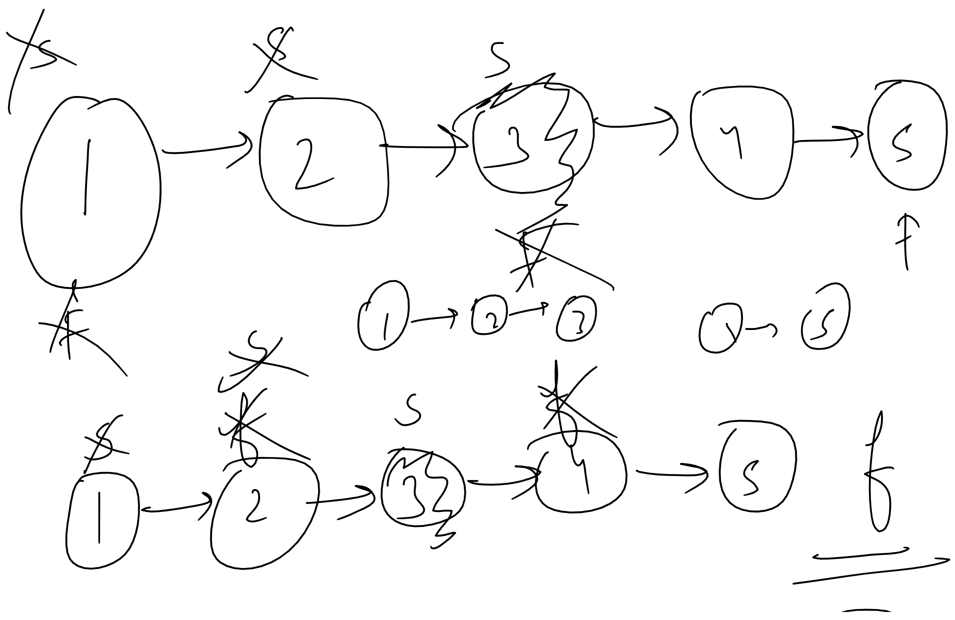
Medium Accuracy: **74.76%** Submissions: **49827** Points: **4**

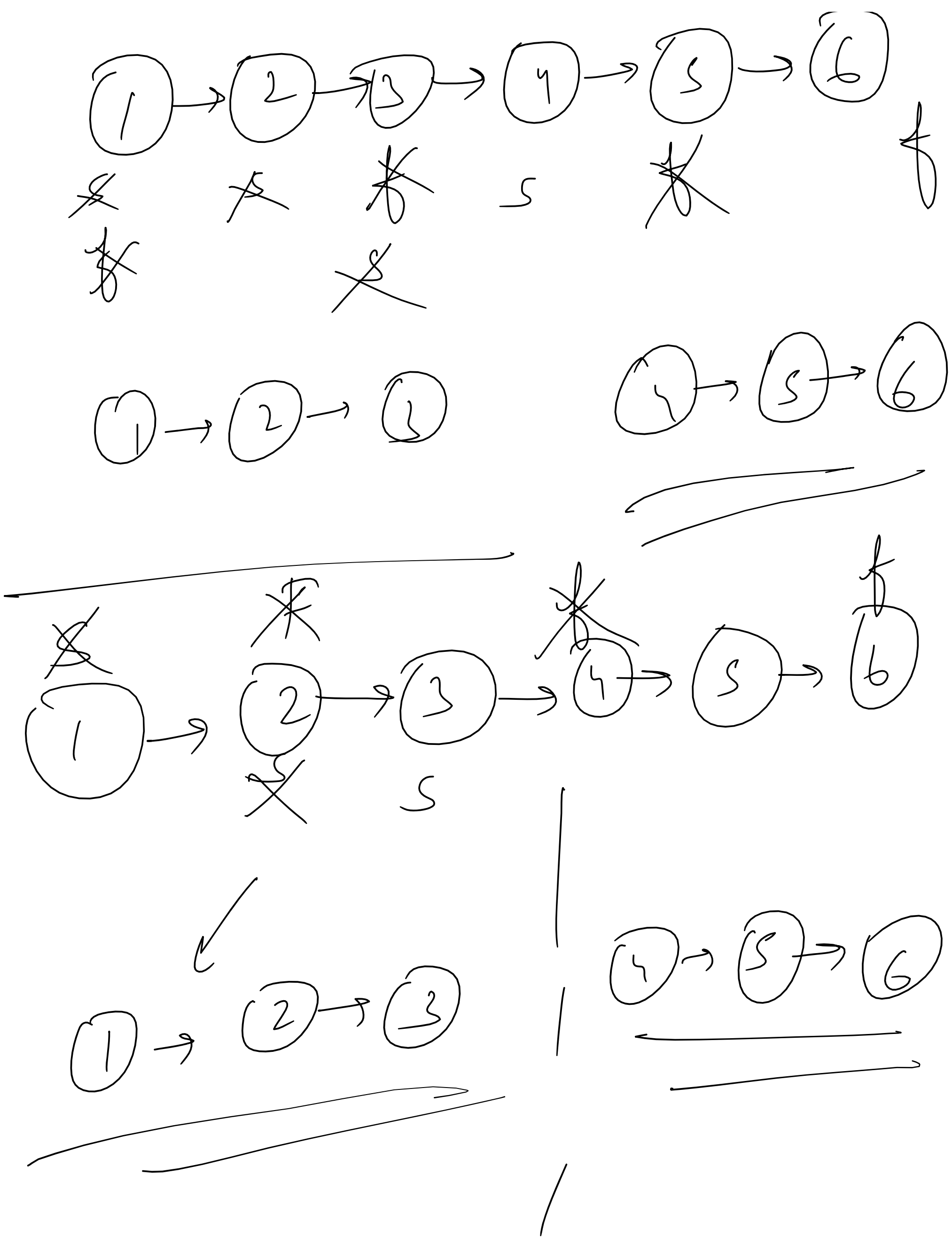
Given Pointer/Reference to the head of the linked list, the task is to **Sort the given linked list using Merge Sort**.

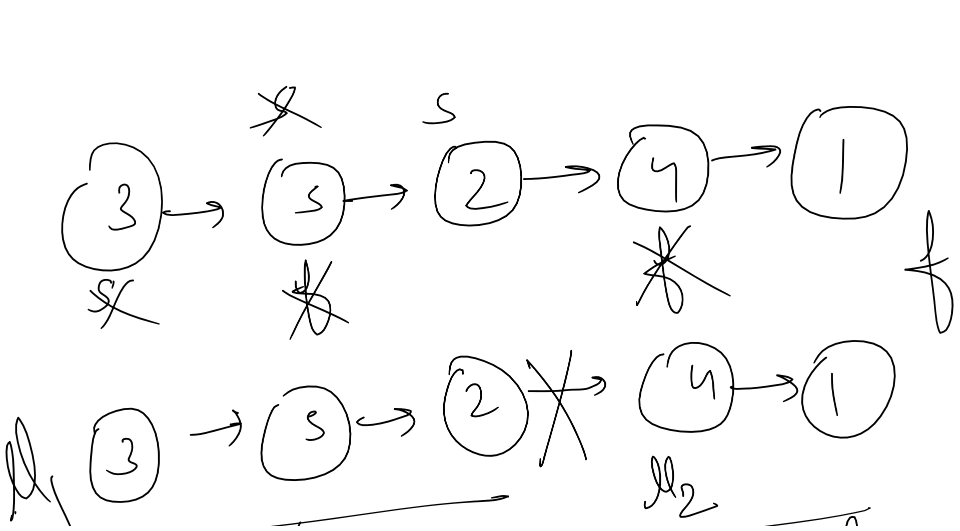
Note: If the length of linked list is odd, then the extra node should go in the first list while splitting.

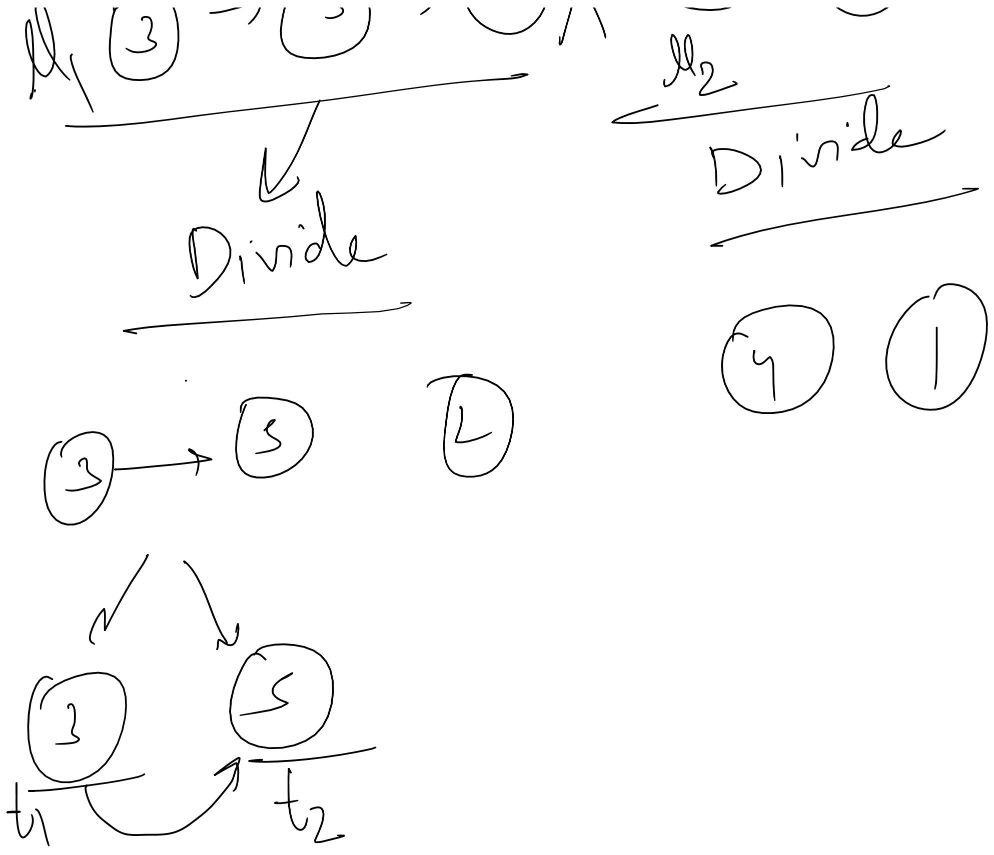
Example 1:

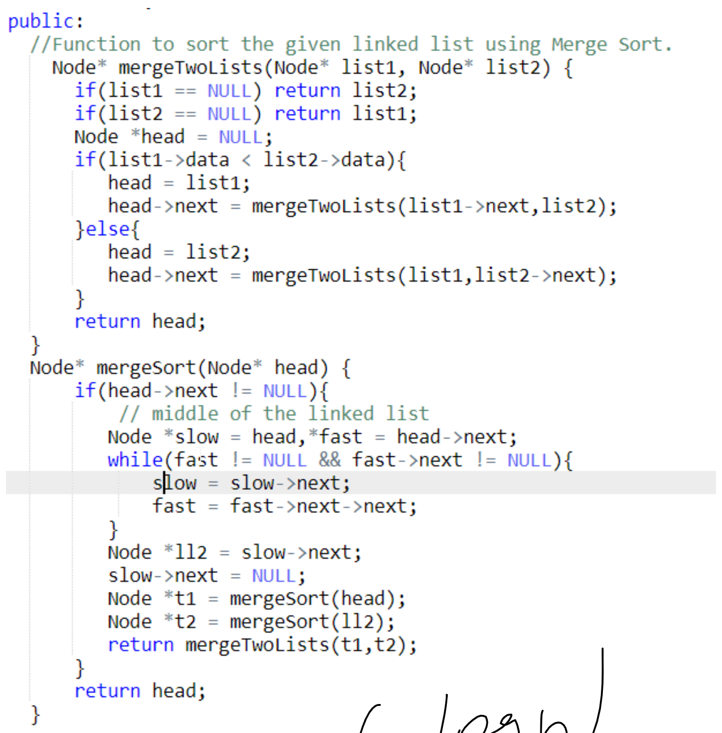
```
Input:
N = 5
value[]  = {3,5,2,4,1}
Output: 1 2 3 4 5
Explanation: After sorting the given
linked list, the resultant matrix
will be 1->2->3->4->5.
```




s
f
s


s
f
ll_1
ll_2





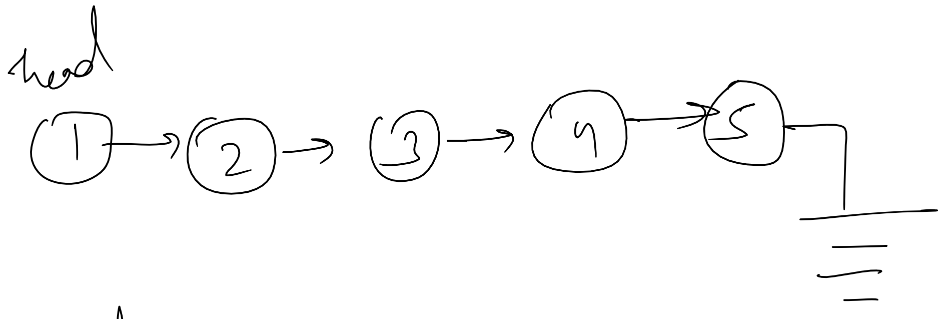
```
public:
  //Function to sort the given linked list using Merge Sort.
    Node* mergeTwoLists(Node* list1, Node* list2) {
      if(list1 == NULL) return list2;
      if(list2 == NULL) return list1;
      Node *head = NULL;
      if(list1->data < list2->data){
         head = list1;
         head->next = mergeTwoLists(list1->next,list2);
      }else{
         head = list2;
         head->next = mergeTwoLists(list1,list2->next);
      }
      return head;
  }
  Node* mergeSort(Node* head) {
      if(head->next != NULL){
          // middle of the linked list
         Node *slow = head,*fast = head->next;
         while(fast != NULL && fast->next != NULL){
             slow = slow->next;
             fast = fast->next->next;
         }
         Node *ll2 = slow->next;
         slow->next = NULL;
         Node *t1 = mergeSort(head);
         Node *t2 = mergeSort(ll2);
         return mergeTwoLists(t1,t2);
      }
      return head;
  }
```

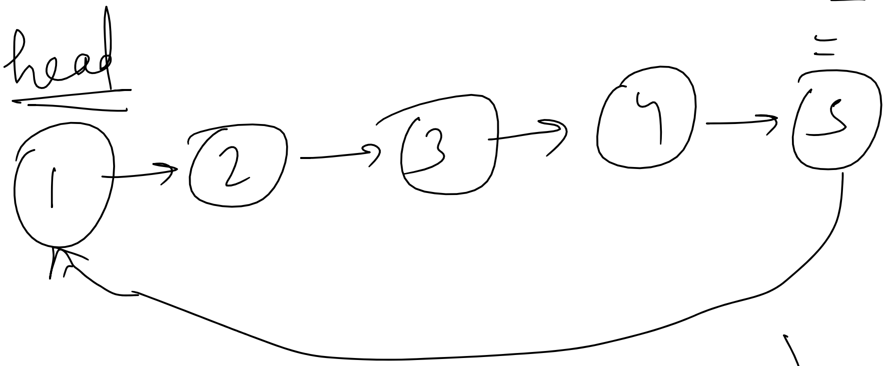
T.C - O(n log n)

S.C - O(n) $\rightarrow$ Recursive Stack Space

Linked list

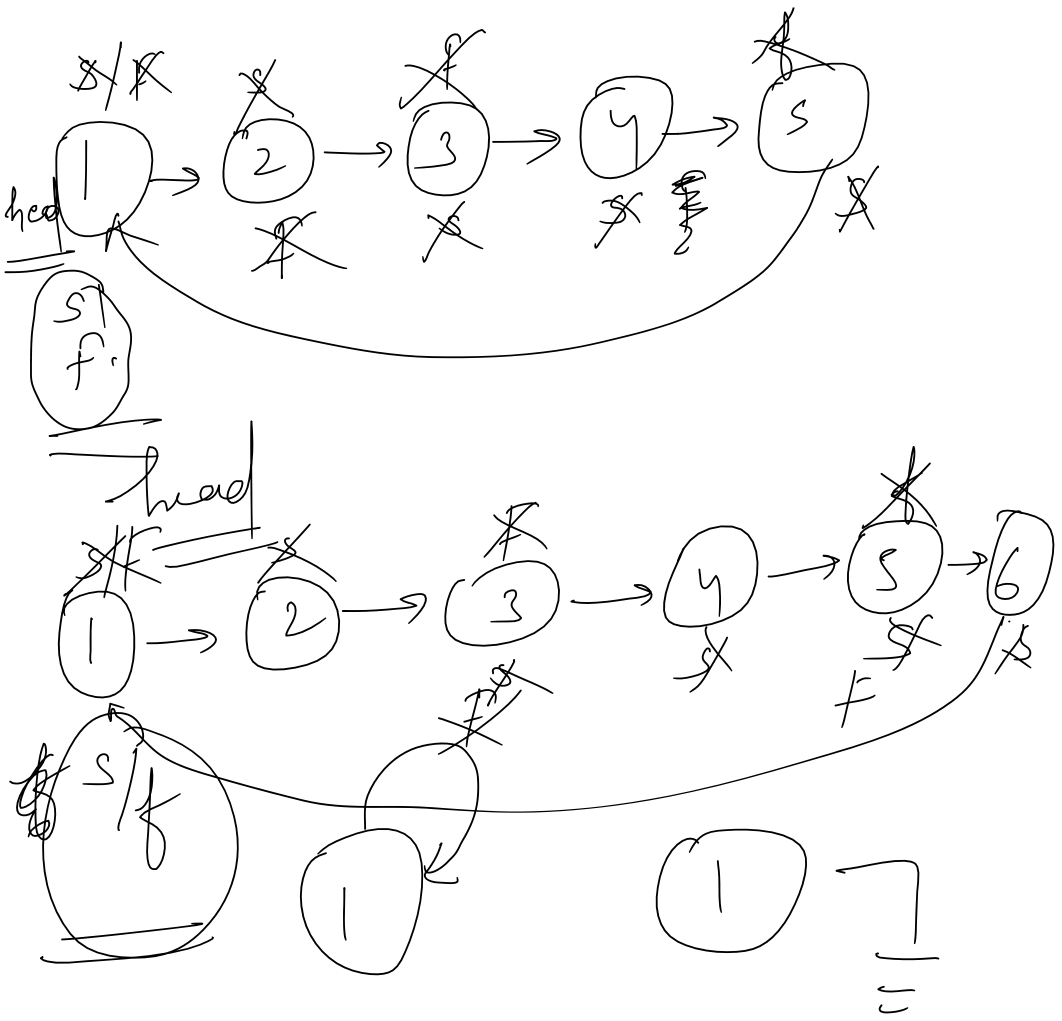
middle of Linked List
Merge two Sorted lists
Merge sort for Linked Lists

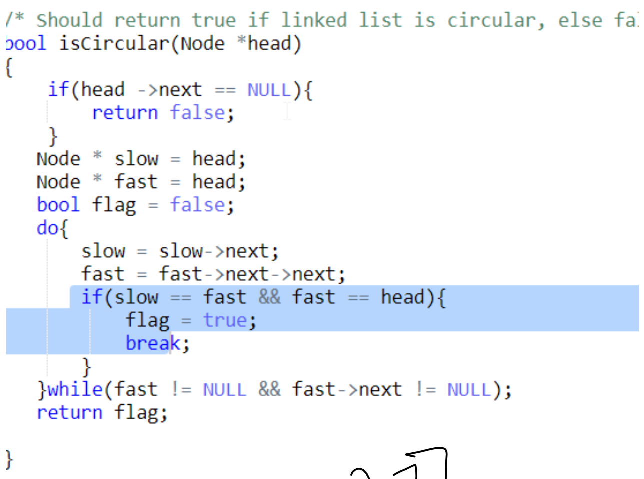




last vali node head को point करने लगा जाए तो इसे कहते हैं ।।

Circular Linked list





```
/* Should return true if linked list is circular, else fa
ool isCircular(Node *head)
{
    if(head ->next == NULL){
        return false;
    }
   Node * slow = head;
   Node * fast = head;
   bool flag = false;
   do{
       slow = slow->next;
       fast = fast->next->next;
       if(slow == fast && fast == head){
           flag = true;
           break;
       }
   }while(fast != NULL && fast->next != NULL);
   return flag;

}
```

T.C - O(N)
S.C - O(1)