|  |  |  |
| --- | --- | --- |
| Sl No | Problem Name | Merge Two Sorted Linked List |
| 1 | Problem No | 21 |
| Problem Link | <https://leetcode.com/problems/merge-two-sorted-lists/> |
| Problem Solution | /\*\*   \* Definition for singly-linked list.   \* public class ListNode {   \*     int val;   \*     ListNode next;   \*     ListNode() {}   \*     ListNode(int val) { this.val = val; }   \*     ListNode(int val, ListNode next) { this.val = val; this.next = next; }   \* }   \*/  class Solution {      public ListNode mergeTwoLists(ListNode l1, ListNode l2) {         ListNode headMerged=new ListNode(0);         ListNode currentNode=headMerged;         while(l1!=null && l2!=null)         {             if(l1.val<=l2.val){             currentNode.next=l1;             l1=l1.next;             }else{                 currentNode.next=l2;                 l2=l2.next;             }             currentNode=currentNode.next;         }         /\*If the linked lists are of unequal sizes \*/         if(l1!=null)         {             currentNode.next=l1;             l1=l1.next;         }         else if(l2!=null)         {             currentNode.next=l2;             l2=l2.next;         }         return headMerged.next;        }  } |

|  |  |  |
| --- | --- | --- |
| Sl No | Problem Name | Remove Nth Node From End Of The List |
| 2 | Problem No | 19 |
| Problem Link | <https://leetcode.com/problems/remove-nth-node-from-end-of-list/> |
| Problem Solution | /\*\*   \* Definition for singly-linked list.   \* public class ListNode {   \*     int val;   \*     ListNode next;   \*     ListNode() {}   \*     ListNode(int val) { this.val = val; }   \*     ListNode(int val, ListNode next) { this.val = val; this.next = next; }   \* }   \*/  class Solution {      public ListNode removeNthFromEnd(ListNode head, int n) {  ListNode dummyHead=new ListNode(0);  dummyHead.next=head;         ListNode slow=dummyHead;         ListNode fast=dummyHead;         for(int i=1;i<=n+1;i++)         {             fast=fast.next;         }         while(fast!=null)         {             slow=slow.next;             fast=fast.next;           }         slow.next=slow.next.next;         return dummyHead.next;      }  } |

|  |  |  |
| --- | --- | --- |
| Sl No | Problem Name | Delete Node in a Linked List |
| 3 | Problem No | 237 |
| Problem Link | <https://leetcode.com/problems/delete-node-in-a-linked-list/> |
| Problem Solution | /\*\*   \* Definition for singly-linked list.   \* public class ListNode {   \*     int val;   \*     ListNode next;   \*     ListNode(int x) { val = x; }   \* }   \*/  class Solution {      public void deleteNode(ListNode node) {          node.val=node.next.val;          node.next=node.next.next;        }  } |

|  |  |  |
| --- | --- | --- |
| Sl No | Problem Name | Remove Duplicates From A Sorted List |
| 4 | Problem No | 83 |
| Problem Link | <https://leetcode.com/problems/remove-duplicates-from-sorted-list/> |
| Problem Solution | /\*\*   \* Definition for singly-linked list.   \* public class ListNode {   \*     int val;   \*     ListNode next;   \*     ListNode() {}   \*     ListNode(int val) { this.val = val; }   \*     ListNode(int val, ListNode next) { this.val = val; this.next = next; }   \* }   \*/  class Solution {      public ListNode deleteDuplicates(ListNode head) {          ListNode tempNode=head;          while(tempNode!=null && tempNode.next!=null)          {              if(tempNode.val==tempNode.next.val)              {                  tempNode.next=tempNode.next.next;              }              else              tempNode=tempNode.next;          }          return head;        }  } |

|  |  |  |
| --- | --- | --- |
| Sl No | Problem Name | Reverse A Linked List |
| 5 | Problem No | 206 |
| Problem Link | <https://leetcode.com/problems/reverse-linked-list/> |
| Problem Solution | /\*\*   \* Definition for singly-linked list.   \* public class ListNode {   \*     int val;   \*     ListNode next;   \*     ListNode() {}   \*     ListNode(int val) { this.val = val; }   \*     ListNode(int val, ListNode next) { this.val = val; this.next = next; }   \* }   \*/  class Solution {      public ListNode reverseList(ListNode head) {          if(head==null)          return head;          ListNode currentNode=head;          ListNode previousNode=null;          ListNode nextNode=head.next;          while(currentNode!=null)          {              currentNode.next=previousNode;              previousNode=currentNode;              currentNode=nextNode;              if(nextNode!=null)              {                  nextNode=nextNode.next;              }            }          return previousNode;        }  } |