|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Problem Link** | | <https://leetcode.com/problems/linked-list-cycle/> | | | |
| **Problem No** | 141 | **Problem Name** | Linked List Cycle | **Difficulty** | Easy |

|  |
| --- |
| /\*\*   \* Definition for singly-linked list.   \* class ListNode {   \*     int val;   \*     ListNode next;   \*     ListNode(int x) {   \*         val = x;   \*         next = null;   \*     }   \* }   \*/  public class Solution {      public boolean hasCycle(ListNode head) {          if(head==null)          {              return false;          }            else          {              ListNode slow=head;              ListNode fast=head;              while(fast!=null && fast.next!=null)              {                  slow=slow.next;                  fast=fast.next.next;                  if(slow==fast)                  {                      return true;                  }              }              return false;            }        }  } |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Problem Link** | | <https://leetcode.com/problems/remove-nth-node-from-end-of-list/> | | | |
| **Problem No** | 19 | **Problem Name** | Remove Nth Node From End Of The Lists | **Difficulty** | Medium |

|  |
| --- |
| /\*\*   \* 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;      }  } |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Problem Link** | | <https://leetcode.com/problems/reverse-linked-list/> | | | |
| **Problem No** | 206 | **Problem Name** | Reverse Linked List | **Difficulty** | Easy |

|  |
| --- |
| /\*\*   \* 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;        }  } |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Problem Link** | | <https://leetcode.com/problems/remove-duplicates-from-sorted-list/> | | | |
| **Problem No** | 83 | **Problem Name** | Remove Duplicates From Sorted List | **Difficulty** | Easy |

|  |
| --- |
| /\*\*   \* 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;        }  } |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Problem Link** | | <https://leetcode.com/problems/delete-node-in-a-linked-list/> | | | |
| **Problem No** | 237 | **Problem Name** | Delete Node In A Linked List | **Difficulty** | Medium |

|  |
| --- |
| /\*\*   \* 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;      }  } |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Problem Link** | | <https://leetcode.com/problems/merge-two-sorted-lists/> | | | |
| **Problem No** | 21 | **Problem Name** | Merge Two Sorted Lists | **Difficulty** | Easy |

|  |
| --- |
| /\*\*   \* 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;        }  } |