Assignment No 10

Ques.1. Answer :

```
class Solution {
    public void deleteNode(ListNode node) {
        ListNode temp =new ListNode();
        while(node.next !=null)       {
            node.val = node.next.val;
             temp = node;
            node= node.next;
        }
        temp.next = null;
    }
}
```

Leetcode Link: https://leetcode.com/problems/delete-node-in-a-linked-list/submissions/

Ques.2. Answer :

```
class Solution {
    public ListNode removeElements(ListNode head, int val) {
          ListNode sentinel = new ListNode();
    sentinel.next = head;

    ListNode prev = sentinel, curr = head;

    while (curr != null)
    {
        if (curr.val == val) prev.next = curr.next;
        else prev = curr;

        curr = curr.next;
    }
    return sentinel.next;

    }
```

```
}
```

Ques.3. Answer :

```
class Solution {
   public ListNode mergeTwoLists(ListNode list1, ListNode list2) {

     if(list1!=null && list2!=null){
      if(list1.val<list2.val){
         list1.next=mergeTwoLists(list1.next,list2);
         return list1;
         }
         else{
            list2.next=mergeTwoLists(list1,list2.next);
            return list2;
      }
      }
      if(list1==null)
         return list2;
      return list1;
   }
}
```

Ques.4. Answer :

```
public class Solution {
   public ListNode detectCycle(ListNode head) {
       ListNode fast = head;
      ListNode slow = head;
      while (fast != null && fast.next != null) {
         slow = slow.next;
         fast = fast.next.next;
         if (fast == slow) {
            slow = head;
            while (slow != fast) {
               slow = slow.next;
               fast = fast.next;
            }
            return slow;
         }
      }
      return null;

   }
}
```

Ques.5. Answer : class Solution {

```
    public ListNode removeNthFromEnd(ListNode head, int n) {
        if(head == null)
            return null;
        ListNode fast = head;
        ListNode slow = head;
        for(int i=0; i<n; i++)
        {
            fast = fast.next;
        }
        //if remove the first node
        if(fast == null)
        {
            head = head.next;
            return head;
        }
        while(fast.next != null)
        {
            fast = fast.next;
            slow = slow.next;
        }
        slow.next = slow.next.next;
        return head;
    }
}
```

Ques.6. Answer :

```
class ListNode {
    int val;
    ListNode next;

    ListNode(int val) {
        this.val = val;
    }
}

public class RotateLinkedList {

    public static ListNode rotateLeft(ListNode head, int k) {
        if (head == null || k <= 0) {
            return head;
        }

        int length = 1;
        ListNode tail = head;

        // Find the length and tail of the linked list
        while (tail.next != null) {
            length++;
            tail = tail.next;
        }

        k = k % length;
        if (k == 0) {
            return head;
        }

        tail.next = head;

        // Find the new head (k-th node from the beginning)
        for (int i = 0; i < length - k; i++) {
            tail = tail.next;
        }

        // Set the new head and break the cycle
        head = tail.next;
        tail.next = null;
```

```
        return head;
    }

    public static void printList(ListNode head) {
        ListNode current = head;
        while (current != null) {
            System.out.print(current.val + " ");
            current = current.next;
        }
    }

    public static void main(String[] args) {
        ListNode head = new ListNode(2);
        head.next = new ListNode(4);
        head.next.next = new ListNode(7);
        head.next.next.next = new ListNode(8);
        head.next.next.next.next = new ListNode(9);

        int k = 3;
        System.out.println("Original List:");
        printList(head);

        head = rotateLeft(head, k);
        System.out.println("\nRotated List:");
        printList(head);
    }
}
```

Ques.7. Answer :

```
class ListNode {
    int val;
    ListNode next;

    ListNode(int val) {
        this.val = val;
    }
}

public class DeleteZeroSumSublists {

    public static ListNode deleteZeroSumSublists(ListNode head) {
        ListNode dummy = new ListNode(0);
        dummy.next = head;
        ListNode current = dummy;

        while (current != null) {
            int sum = 0;
            ListNode runner = current.next;

            while (runner != null) {
                sum += runner.val;

                if (sum == 0) {
                    current.next = runner.next;
                    break;
                }

                runner = runner.next;
            }

            if (runner == null) {
                current = current.next;
            }
        }

        return dummy.next;
    }
```

```
    public static ListNode createLinkedList(int[] values) {
        ListNode dummy = new ListNode(0);
        ListNode current = dummy;

        for (int val : values) {
            current.next = new ListNode(val);
            current = current.next;
        }

        return dummy.next;
    }

    public static void printList(ListNode head) {
        ListNode current = head;
        while (current != null) {
            System.out.print(current.val + " ");
            current = current.next;
        }
    }

    public static void main(String[] args) {
        int[] values = {1, 2, -3, 3, 1};
        ListNode head = createLinkedList(values);

        System.out.println("Original List:");
        printList(head);

        head = deleteZeroSumSublists(head);
        System.out.println("\nFinal List:");
        printList(head);
    }
}
```

Ques.8. Answer :

```
class ListNode {
    int val;
    ListNode next;

    ListNode(int val) {
        this.val = val;
    }
}

public class ReorderLinkedList {

    public static ListNode reorderList(ListNode head) {
        if (head == null || head.next == null) {
            return head;
        }

        ListNode oddHead = head;
        ListNode evenHead = head.next;
        ListNode oddCurrent = oddHead;
        ListNode evenCurrent = evenHead;

        while (evenCurrent != null && evenCurrent.next != null) {
            oddCurrent.next = evenCurrent.next;
            oddCurrent = oddCurrent.next;
            evenCurrent.next = oddCurrent.next;
            evenCurrent = evenCurrent.next;
        }

        oddCurrent.next = evenHead;
        return head;
    }

    // Helper method to create a linked list from an array
    public static ListNode createLinkedList(int[] values) {
        ListNode dummy = new ListNode(0);
        ListNode current = dummy;

        for (int val : values) {
```

```
            current.next = new ListNode(val);
            current = current.next;
        }

        return dummy.next;
    }

    // Helper method to print the linked list
    public static void printList(ListNode head) {
        ListNode current = head;
        while (current != null) {
            System.out.print(current.val + " ");
            current = current.next;
        }
    }

    public static void main(String[] args) {
        int[] values = {1, 2, 3, 4, 5};
        ListNode head = createLinkedList(values);

        System.out.println("Original List:");
        printList(head);

        head = reorderList(head);
        System.out.println("\nReordered List:");
        printList(head);
    }
}
```