1. [Delete Node in a Linked List](https://leetcode.com/problems/delete-node-in-a-linked-list/)
2. [Remove Linked List Elements](https://leetcode.com/problems/remove-linked-list-elements/)

public class LinkedList{

    public static class Node{

        int data;

        Node next;

        public Node(int data){

            this.data= data;

            this.next=null;

        }

    }

    public static Node head;

    public static Node tail;

    public static int size;

    //Methods for insertion

    public synchronized void insertAtBegin(int data){

        Node newNode= new Node(data);//created new node

        size++;

        if(head==null){

            head=tail=newNode;

            return;

        }

        newNode.next=head;//linked nodes

        head=newNode;//head position changed

    }

    //Finding middle term using two pointer approach

    public static int middleTerm(){

        Node fast=head,slow=head;

        while(fast!=null && fast.next!=null){

            slow=slow.next;

            fast=fast.next.next;

        }

        return slow.data;

    }

    public synchronized void insertAtLast(int data){

        Node newNode= new Node(data);

        size++;

        if(head==null){

            head=tail=newNode;

            return;

        }

        tail.next=newNode;

        tail=newNode;

    }

    public synchronized void insertionAtMiddle(int data,int index){

        if(index==0){

            insertAtBegin(data);

            return;

        }

        Node newNode= new Node(data);

        size++;

        Node temp=head;

        int i=0;

        while(i<(index-1)){

            temp=temp.next;

            i++;

        }

        newNode.next=temp.next;

        temp.next=newNode;

    }

    //Method to Display Linked list

    public void Display(){

       if(head==null){

        System.out.println("Linked List is Empty");

        return;

       }

        Node temp=head;

        while(temp!=null){

            System.out.print(" "+temp.data);

            temp=temp.next;

        }System.out.println();

    }

    //Method for Deletion

    public synchronized void deletionAtBegin(){

       if(size==0){

        System.out.println("Linked List is already empty");

        return ;

       }

       if(size==1){

        head=tail=null;

        size=0;

        return;

       }

        head=head.next;

        size--;

    }

    public synchronized void deletionAtLast(){

        if(size==0){

            System.out.println("Linked List is already empty");

            return ;

           }

           if(size==1){

            head=tail=null;

            size=0;

            return;

           }

           Node temp=head;;

           while(temp.next!=tail){

            temp=temp.next;

           }

        tail=temp;

        tail.next=null;

        size--;

    }

    public synchronized void deletionAtMiddle(int index){

        if(size==0){

            System.out.println("Linked List is already empty");

            return ;

           }

           if(size==1){

            head=tail=null;

            size=0;

            return;

           }

           Node temp=head;

           int i=0;

           while(i<(index-1)){

            temp=temp.next;

            i++;

           }

           temp.next=temp.next.next;

    }

    public static void main(String ar[]){

        LinkedList ll = new LinkedList();

        ll.Display();

        ll.insertAtBegin(1);

        ll.insertAtBegin(2 );

        ll.insertAtLast(4);

        ll.insertAtLast(5);

        ll.Display();

        ll.deletionAtBegin();

        ll.Display();

        ll.insertionAtMiddle(45,2);

        ll.Display();

        System.out.println(" "+size);

        ll.deletionAtLast();

        ll.Display();

        ll.deletionAtMiddle(1);

        ll.Display();

    }

}

1. [Merge Two Sorted List](https://leetcode.com/problems/merge-two-sorted-lists/)

public class mergeTwoSortedList {

    public static void main(String[] args) {

        int arr1[]={1,2,3,4,5};

        int arr2[]={6,7,8,9,10};

        int arr3[]=new int[arr1.length+arr2.length];

        int i=0,j=0,k=0;

        while(i<arr1.length && j<arr2.length){

            if(arr1[i]<arr2[j]){

                arr3[k]=arr1[i];

                i++;

                k++;

            }

            else{

                arr3[k]=arr2[j];

                j++;

                k++;

            }

        }

        while(i<arr1.length){

            arr3[k]=arr1[i];

            i++;

            k++;

        }

        while(j<arr2.length){

            arr3[k]=arr2[j];

            j++;

            k++;

        }

        for(int l=0;l<arr3.length;l++){

            System.out.print(arr3[l]+" ");

        }

    }

}

1. [Linked List Cycle](https://leetcode.com/problems/linked-list-cycle-ii/)

package Assignment\_10;

public class linkListCycle {

    public class Solution

{

public ListNode detectCycle(ListNode head)

{

ListNode curr = head;

ListNode []arr =new ListNode[10000];

int ind=0;

while(curr!=null)

{

arr[ind]=curr;

curr=curr.next;

ind++;

for(int i=0;i<ind;i++)

{

if(arr[i]==curr)

{

return curr;

}

}

}

return curr;

}

}

}

1. [Remove the Nth node from the linked list](https://leetcode.com/problems/remove-nth-node-from-end-of-list/)

public class removeNnodeFromLinkedList {

class Solution {

    public ListNode removeNthFromEnd(ListNode head, int n) {

        ListNode slow = head,fast = head;

        for(int i =0;i<n;i++){

            fast = fast.next;

        }

//agar mera fast ka value null hojata hai toh hum hum head ke element of delete kar dete hai.....

        if(fast==null){return head.next;}

        while(fast.next!=null){

            slow = slow.next;

            fast = fast.next;

        }

        slow.next = slow.next.next;

        return head;

    }

}

}

1. Given a singly linked list of size N. The task is to left-shift the linked list by k nodes, where k is a given positive integer smaller than or equal to the length of the linked list.

Input: N = 5

value[] = {2, 4, 7, 8, 9}

k = 3

Output: 8 9 2 4 7

Explanation:Rotate 1:4 -> 7 -> 8 -> 9 -> 2

Rotate 2: 7 -> 8 -> 9 -> 2 -> 4

Rotate 3: 8 -> 9 -> 2 -> 4 -> 7

public class question6 {

    public static void main(String[] args) {

        int arr[]={2,4,7,8,9};

        int k=3;

        int arr1[]=new int[arr.length];

        int i=0;

        while(i<k){

            arr1[i]=arr[arr.length-k+i];

            i++;

        }

        int j=0;

        while(i<arr.length){

            arr1[i]=arr[j];

            i++;

            j++;

        }

        for(int l=0;l<arr1.length;l++){

            System.out.print(arr1[l]+" ");

        }

    }

}

1. Given the head of a linked list, we repeatedly delete consecutive sequences of nodes that sum to 0 until there are no such sequences.

After doing so, return the head of the final linked list.  You may return any such answer.

(Note that in the examples below, all sequences are serializations of ListNode objects.)

Input: head = [1,2,-3,3,1]

Output: [3,1]

Note: The answer [1,2,1] would also be accepted.

//Not able to understand fully(looked over internet leetcode and try brute force approach)

import java.util.Arrays;

public class question7 {

public static ListNode removeZeroSumSublists(ListNode head) {

    ListNode node = head;

    int arr[] = new int[1000];

    int size=0;

    while(node != null){

        arr[size] = node.val;

        node = node.next;

        size++;

    }

    for(int i=0; i<size; i++){

        int sum = 0;

        for(int j=i; j<size; j++){

            sum += arr[j];

            if(sum == 0){

                Arrays.fill(arr, i, j+1, 0);

                i = j;

            }

        }

    }

    for(int i=0; i<size; i++){

        System.out.println(arr[i]);

    }

    head = new ListNode(0);

    ListNode tail = head;

    for(int i=0; i<size; i++){

        if(arr[i] != 0){

            node = new ListNode(arr[i]);

            tail.next = node;

            tail = node;

        }

    }

    tail.next = null;

    return head.next;

}

}