SDE sheet Questions (Revise these if you have very less time)

Problem
Reverse a LinkedList
Find the middle of LinkedList
Merge two sorted Linked List (use method used in mergeSort)
Remove N-th node from back of LinkedList
Add two numbers as LinkedList
Delete a given Node when a node is given.(O(1) solution)

Problem
Find intersection point of Y LinkedList
Detect a cycle in Linked List
Reverse a LinkedList in groups of size k.
Check if a LinkedList is palindrome or not.
Find the starting point of the Loop of LinkedList
Flattening of a LinkedList

Rotate a LinkedList
Clone a Linked List with random and next pointer

LinkedList	Write a Program to reverse the Linked List. (Both Iterative and recursive)
LinkedList	Reverse a Linked List in group of Given Size. [Very Imp]
LinkedList	Write a program to Detect loop in a linked list.
LinkedList	Write a program to Delete loop in a linked list.
LinkedList	Find the starting point of the loop.
LinkedList	Remove Duplicates in a sorted Linked List.
LinkedList	Remove Duplicates in a Un-sorted Linked List.
LinkedList	Write a Program to Move the last element to Front in a Linked List.
LinkedList	Add “1” to a number represented as a Linked List.
LinkedList	Add two numbers represented by linked lists.
LinkedList	Intersection of two Sorted Linked List.
LinkedList	Intersection Point of two Linked Lists.
LinkedList	Merge Sort For Linked lists.[Very Important]
LinkedList	Quicksort for Linked Lists.[Very Important]

LinkedList	Find the middle Element of a linked list.
LinkedList	Check if a linked list is a circular linked list.
LinkedList	Split a Circular linked list into two halves.
LinkedList	Write a Program to check whether the Singly Linked list is a palindrome or not.
LinkedList	Deletion from a Circular Linked List.
LinkedList	Reverse a Doubly Linked list.
LinkedList	Find pairs with a given sum in a DLL.
LinkedList	Count triplets in a sorted DLL whose sum is equal to given value “X”.
LinkedList	Sort a “k”sorted Doubly Linked list.[Very IMP]
LinkedList	Rotate DoublyLinked list by N nodes.
LinkedList	Rotate a Doubly Linked list in group of Given Size.[Very IMP]
LinkedList	Can we reverse a linked list in less than O(n) ?
LinkedList	Why Quicksort is preferred for. Arrays and Merge Sort for LinkedLists ?
LinkedList	Flatten a Linked List
LinkedList	Sort a LL of 0's, 1's and 2's

LinkedList	Clone a linked list with next and random pointer
LinkedList	Merge K sorted Linked list
LinkedList	Multiply 2 no. represented by LL
LinkedList	Delete nodes which have a greater value on right side
LinkedList	Segregate even and odd nodes in a Linked List
LinkedList	Program for n’th node from the end of a Linked List
LinkedList	Find the first non-repeating character from a stream of characters

Various sheets

Segregate Even And Odd Nodes In A Linkedlist

Easy

Given a singly linklist, modify the list such that all the even numbers appear before all the odd numbers in the modified list. The order of appearance of numbers within each segregation should be same as that in the original list.

Constraints

`0 <= N <= 10^6`

Format

Input

1->7->2->6->3->5->4->null

Output

2->6->4->1->7->3->5->null

```java
public static ListNode segregateEvenOdd(ListNode head) {
    ListNode even=new ListNode(-1);
    ListNode h1=even;
    ListNode odd=new ListNode(-1);
    ListNode h2=odd;
    ListNode curr=head;
    while(curr!=null){
        if(curr.val%2==0){
            even.next=curr;
            curr=curr.next;
            even=even.next;
        }
        else{
            odd.next=curr;
            curr=curr.next;
            odd=odd.next;

        }
    }
    even.next=h2.next;
    odd.next=null;
    return h1.next;
}
```

Easy Question

Problem Statement Suggest Edit

You have been given a Singly Linked List of integers, determine if it forms a cycle or not. If there is a cycle, remove the cycle and return the list.

A cycle occurs when a node's 'next' points back to a previous node in the list.

Input Format :

The first line of input contains a single integer T, representing the number of test cases or queries to be run.

The first line of each test case contains the elements of the singly linked list separated by a single space and terminated by -1 and hence -1 would never be a list element.

The second line contains the integer position "pos" which represents the position (0-indexed) in the linked list where the tail connects to. If "pos" is -1, then there is no cycle in the linked list.

Output Format :

For each test case, print two lines.

The first line contains 'True' if the linked list has a cycle, otherwise 'False'.

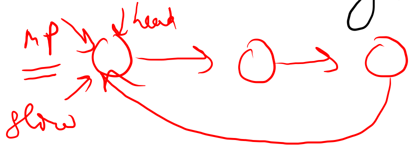
The second line contains the elements of the singly linked list separated by a single space and terminated by -1. Hence, -1 would never be a list element.

```java
}
public static boolean detectAndRemoveCycle(Node head)
{
    if(head==null||head.next==null) return false;
    Node meetingPoint=detectCycle(head);
    if(meetingPoint!=null){
        if(meetingPoint==head){
            Node curr=head;
            while(curr.next!=head){
                curr=curr.next;
            }
            curr.next=null;
            return true;
        }
        Node slow=head;
        while(meetingPoint.next!=slow.next){
            meetingPoint=meetingPoint.next;
            slow=slow.next;
            }
    meetingPoint.next=null;
        return true;
    }
    return false;


}
}
```

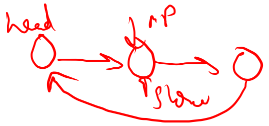
```java
{
    public static Node detectCycle(Node head){
        if(head==null||head.next==null) return null;
        Node slow=head;
        Node fast=head;
        while(fast!=null&&fast.next!=null){
            slow=slow.next;
            fast=fast.next.next;
            if(slow==fast) return slow;
        }
        return null;
    }
```

Detect cycle tak the cycle kha hai Ab delete kaise karna hai
Now we know how to get starting point of loop hme just starting point se pehle jahan hai tak
(meeting Point.next != slow.next) as dono ka next ab starting point of loop ko hai & then do Meeting Point.next = null do. Now see this case

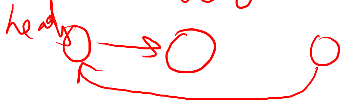


MP → Meeting Point

When starting point of loop is head now if we do that



dono ka next same so ye link toot jaega & LL ho ke



So for this we here separate case

Line 36 to 44 handles that case if MP == head then End tak jao jha curr.next != head na ho jae as LastNode & then uske next Null kr do

Check If Circular Linked List

Basic Accuracy: 50.74% Submissions: 77192 Points: 1

Given **head**, the head of a singly linked list, find if the linked list is circular or not. A linked list is called circular if it not NULL terminated and all nodes are connected in the form of a cycle. An empty linked list is considered as circular.

Note: The linked list does not contains any inner loop.

Example 1:

```
Input:
LinkedList: 1->2->3->4->5
(the first and last node is connected,
i.e. 5 --> 1)
Output: 1
```

values is true, otherwise 0)

Expected Time Complexity: O(N).
Expected Auxiliary Space: O(1).

```java
*/
class GfG
{
    boolean isCircular(Node head)
    {
    // Your code here
    if(head==null) return true;
    Node curr=head;
    while(curr!=null){

        if(curr.next==head) return true;
         curr=curr.next;
    }
    return false;
    }
}
```

Intersection of two sorted Linked lists

Easy Accuracy: 29.44% Submissions: 63196 Points: 2

Given two lists sorted in increasing order, create a new list representing the intersection of the two lists. The new list should be made with its own memory — the original lists should not be changed.
Note: The list elements are not necessarily distinct.

Example 1:

```
Input:
L1 = 1->2->3->4->6
L2 = 2->4->6->8
Output: 2 4 6
Explanation: For the given first two
linked list, 2, 4 and 6 are the elements
in the intersection.
```

Expected Time Complexity : O(n+m)
Expected Auxilliary Space : O(n+m)
Note: n,m are the size of the linked lists.

Constraints:
1 <= size of linked lists <= 5000
1 <= Data in linked list nodes <= 1000

```java
{
    public static Node findIntersection(Node head1, Node head2)
    {
        Node h1=head1;
        Node h2=head2;
        Node dummy=new Node(-1);
        Node res=dummy;
        while(h1!=null&&h2!=null){
            if(h1.data==h2.data){
                Node node=new Node(h1.data);
                res.next=node;
                res=res.next;
                h1=h1.next;
                h2=h2.next;
            }
            else if(h1.data<h2.data){
                h1=h1.next;
            }
            else h2=h2.next;
        }
        return dummy.next;
    }
}
```

Circular & Doubly LL will be done from
here !!