**Lab 5**

Solve these problems for the linked list.

Deadline: Week 7

<https://leetcode.com/problems/delete-node-in-a-linked-list/>

class Solution {

public void deleteNode(ListNode node) {

if(node == null)

return;

if (node.next == null ){

node = null ;

}

node.val = node.next.val ;

node.next = node.next.next;

}

}

<https://leetcode.com/problems/reverse-linked-list/>

class Solution

{

public ListNode reverseList(ListNode head) {

ListNode temp, current, prev;

current = head;

prev = null;

while(current!=null){

temp = current.next;

current.next = prev;

prev = current;

current = temp;

}

head = prev;

return head;

}

}

<https://leetcode.com/problems/merge-two-sorted-lists/>

<https://leetcode.com/problems/linked-list-cycle/>

public class Solution {

public boolean hasCycle(ListNode head) {

ListNode curr=head;

//as limit is 10^-5 to 10^5

int ans=1000000;

while(curr!=null){

if(curr.val==1000000){

return true;}

curr.val=1000000;

curr=curr.next;

}

return false;

}

}

<https://leetcode.com/problems/partition-list/>

class Solution {

public ListNode partition(ListNode head, int x) {

ListNode head1=new ListNode(0);ListNode t1=head1;

ListNode head2=new ListNode(0);ListNode t2=head2;

ListNode t=head;

while(t!=null){

if(t.val<x){

t1.next=new ListNode(t.val);

t1=t1.next;

}else{

t2.next=new ListNode(t.val);

t2=t2.next;

}t=t.next;

}t1.next=head2.next;

return head1.next;

}}

<https://leetcode.com/problems/intersection-of-two-linked-lists/>

public class Solution {

public ListNode getIntersectionNode(ListNode headA, ListNode headB) {

int l1 = 0;

int l2 = 0;

ListNode temp = headA;

while(temp != null){

l1++;

temp = temp.next;

}

temp = headB;

while(temp != null){

l2++;

temp = temp.next;

}

if(l1>l2) for(int i=0;i<l1-l2;i++) headA = headA.next;

else if(l2>l1) for(int i=0;i<l2-l1;i++) headB = headB.next;

while(headA != headB){

headA = headA.next;

headB = headB.next;

}

return headA;

}

}

<https://leetcode.com/problems/palindrome-linked-list/>

class Solution {

public boolean isPalindrome(ListNode head) {

ListNode slow=head;

ListNode fast=head;

ListNode i=head;

ListNode j;

boolean odd=false;

while(slow!=null && fast!=null){

slow=slow.next;

fast=fast.next;

if(fast == null){

odd=true;

break;

}

fast=fast.next;

}

j = reverse(slow);

while(j!=null){

if(j.val != i.val){

return false;

}

i=i.next;

j=j.next;

}

return true;

}

private ListNode reverse(ListNode start){

ListNode prev=null;

ListNode curr=start;

while(curr!=null){

ListNode temp=curr.next;

curr.next=prev;

prev=curr;

curr=temp;

}

return prev;

}

}

<https://leetcode.com/problems/sort-list/>