Function to check if a singly linked list is palindrome

METHOD 1 (Use a Stack)

A simple solution is to use a stack of list nodes. This mainly involves three steps.
1) Traverse the given list from head to tail and push every visited node to stack.
2) Traverse the list again. For every visited node, pop a node from stack and compare data of popped node with currently visited node.
3) If all nodes matched, then return true, else false.

Time complexity of above method is O(n), but it requires O(n) extra space. Following methods solve this with constant extra space.

METHOD 2 (By reversing the list)
This method takes O(n) time and O(1) extra space.
1) Get the middle of the linked list.
2) Reverse the second half of the linked list.
3) Check if the first half and second half are identical.
4) Construct the original linked list by reversing the second half again and attaching it back to the first half

```
public class Solution {
   public int lPalin(ListNode A) {
      ListNode dummy=ReverseList(A);
      Boolean flag=isPalindrome(dummy,A);
      if(flag){
         return 1;
      }
      return 0;
   }
   public ListNode ReverseList(ListNode A){
      ListNode head=null;
      while(A!=null){
         ListNode temp=new ListNode(A.val);
         temp.next=head;
         head=temp;
         A=A.next;
      }
      return head;
   }
   public boolean isPalindrome(ListNode A,ListNode B){
```

```
        while(A!=null && B!=null){
            if(A.val!=B.val){
                return false;
            }
            A=A.next;
            B=B.next;
        }
        return A==null && B==null;
    }
}
```

Remove Duplicates from Sorted List

Asked in: Microsoft VMWare

Suggest Edit | Bookmark

Given a sorted linked list, delete all duplicates such that each element appear only once.

For example,
Given `1->1->2`, return `1->2`.
Given `1->1->2->3->3`, return `1->2->3`.

See Expected Output

Seen this question in a real interview before Yes No

```
public class Solution {
    public ListNode deleteDuplicates(ListNode A) {
        if(A==null || A.next==null){
            return A;
        }
        ListNode cur=A;
        ListNode fwd=A.next;
        while(fwd!=null){
            if(cur.val==fwd.val){
                cur.next=fwd.next;
                fwd=fwd.next;
            }
            else{
                cur=cur.next;
                fwd=cur;
            }
        }
        return A;


    }
}
```

Remove Duplicates from Sorted List II

Asked in: Microsoft VMWare

Suggest Edit | Bookmark

Given a sorted linked list, delete all nodes that have duplicate numbers, leaving only distinct numbers from the original list.

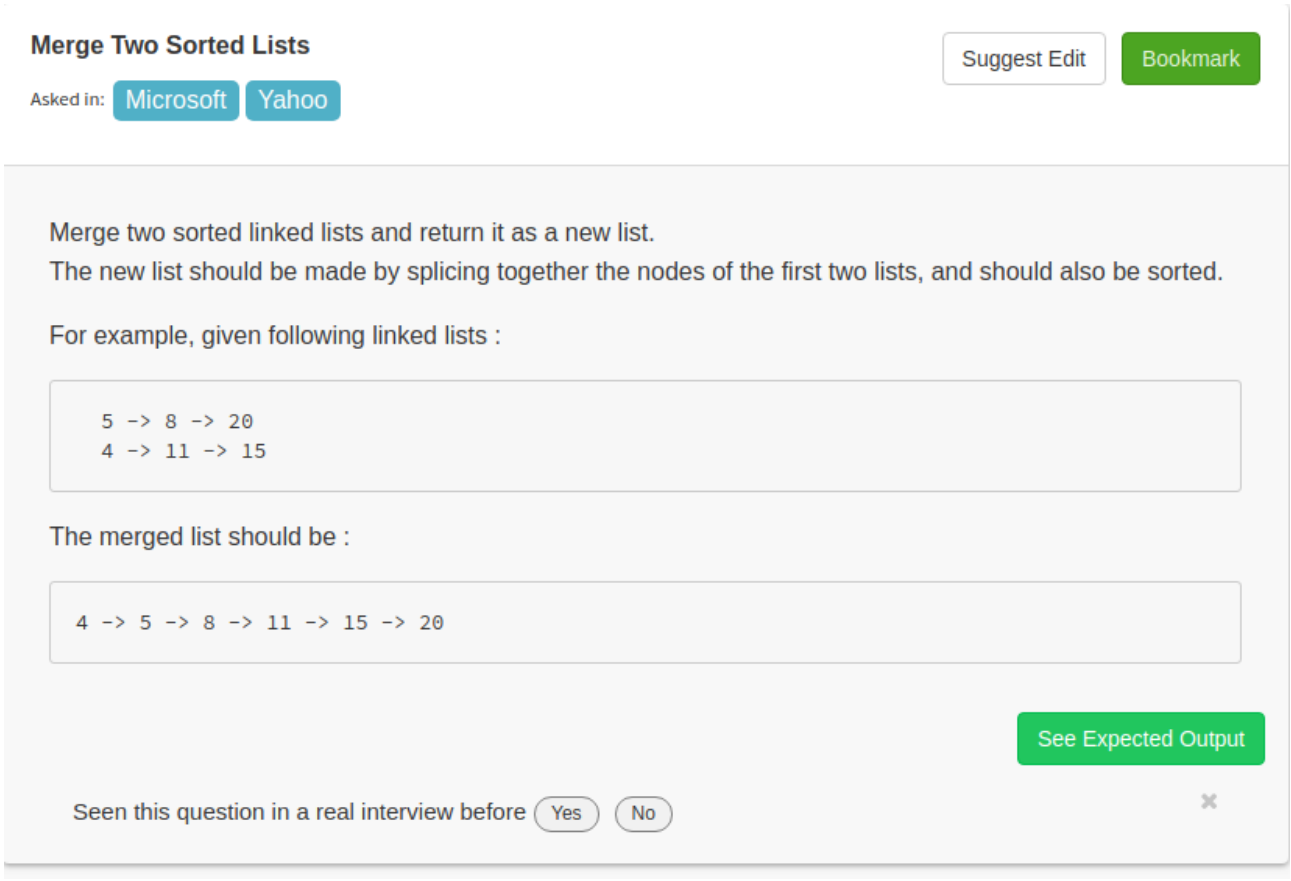
For example,
Given `1->2->3->3->4->4->5`, return `1->2->5`.
Given `1->1->1->2->3`, return `2->3`.

See Expected Output

Seen this question in a real interview before Yes No

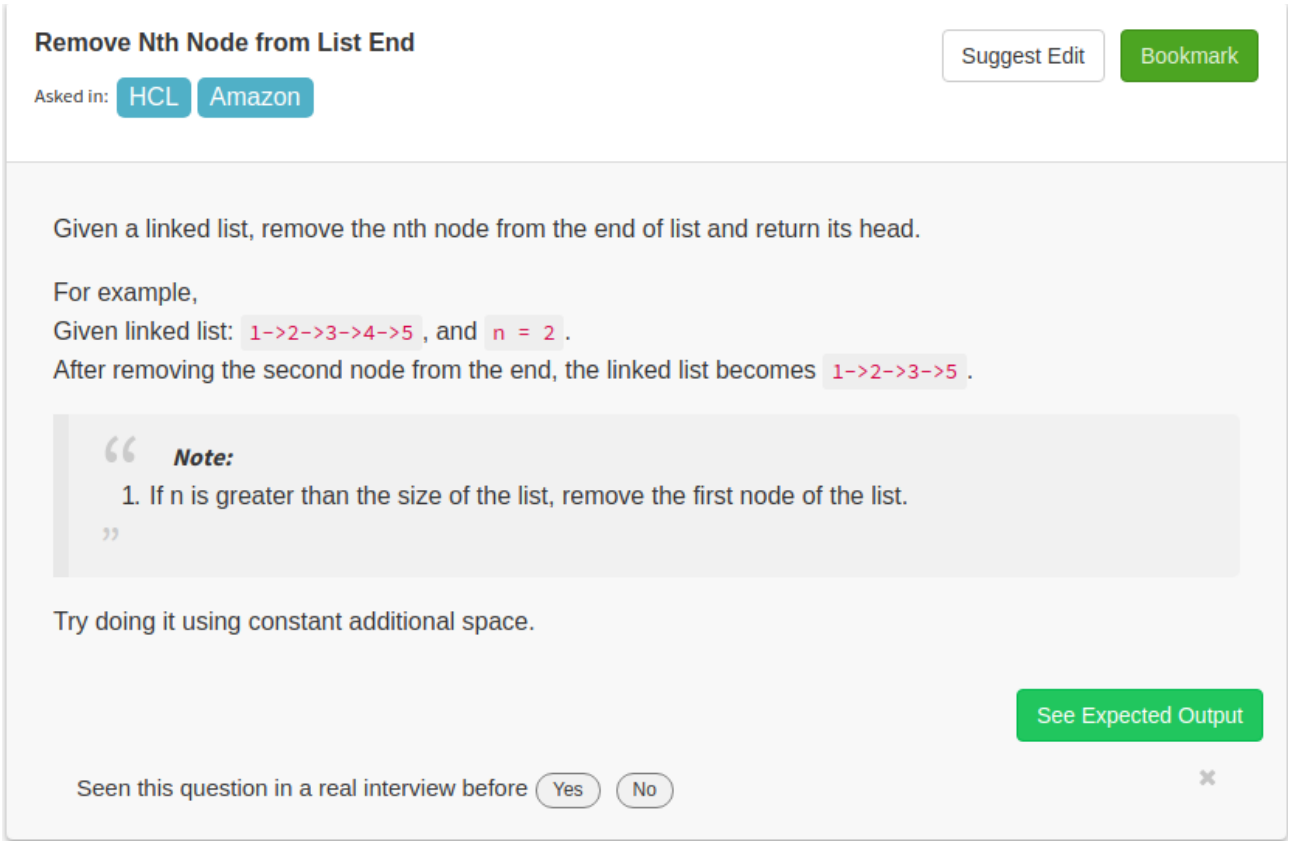
```
/**
 * Definition for singly-linked list.
 * class ListNode {
 *     public int val;
 *     public ListNode next;
 *     ListNode(int x) { val = x; next = null; }
 * }
 */
public class Solution {
    public ListNode deleteDuplicates(ListNode A) {
        if(A.next==null || A==null){
            return A;
        }
        ListNode cur=A;
        ListNode result=new ListNode(0);
        result.next=A;
        ListNode dummy=result;
        while(cur!=null){
            while(cur.next!=null && cur.val==cur.next.val){
                cur=cur.next;
            }
            if(dummy.next==cur){
                dummy=dummy.next;
            }
            else{
                dummy.next=cur.next;
            }
            cur=cur.next;
        }
        return result.next;



    }
```

}




```
/**
 * Definition for singly-linked list.
 * class ListNode {
 *     public int val;
 *     public ListNode next;
 *     ListNode(int x) { val = x; next = null; }
 * }
 */
public class Solution {
    public ListNode mergeTwoLists(ListNode A, ListNode B) {
        ListNode head=new ListNode(0);
        ListNode add=head;
        if(A==null && B==null){
            return null;
        }
        if(A==null){
            return B;
        }
        if(B==null){
            return A;
        }
        while(A!=null && B!=null){
            if(A.val<=B.val){
                add.next=A;
                A=A.next;
            }
```

```
            else{
                add.next=B;
                B=B.next;
            }
            add=add.next;
        }
        if(A!=null){
            add.next=A;
        }
        if(B!=null){
            add.next=B;
        }
        return head.next;


    }
}
```



Remove Nth Node from List End

Asked in: HCL Amazon

Suggest Edit | Bookmark

Given a linked list, remove the nth node from the end of list and return its head.

For example,
Given linked list: `1->2->3->4->5`, and `n = 2`.
After removing the second node from the end, the linked list becomes `1->2->3->5`.

> ***Note:***
> 1. If n is greater than the size of the list, remove the first node of the list.

Try doing it using constant additional space.

See Expected Output

Seen this question in a real interview before Yes No

```
/**
 * Definition for singly-linked list.
 * class ListNode {
 *     public int val;
 *     public ListNode next;
 *     ListNode(int x) { val = x; next = null; }
 * }
 */
public class Solution {
    public ListNode removeNthFromEnd(ListNode A, int B) {
```

```
        if(A==null || A.next==null && B==1){
            return null;
        }
        ListNode dummy=A;
        int n=0;
        while(dummy!=null){
            n++;
            dummy=dummy.next;
        }
        if(B>=n){
            return A.next;
        }
        int k=n-B;
        dummy=A;
        while(k>1){
            dummy=dummy.next;
            k--;

        }
        dummy.next=dummy.next.next;
        return A;


    }

}
```

Rotate List

Asked in: Amazon

Suggest Edit | Bookmark

Given a list, rotate the list to the right by k places, where k is non-negative.

For example:

Given `1->2->3->4->5->NULL` and `k = 2`,
return `4->5->1->2->3->NULL`.

See Expected Output

Seen this question in a real interview before Yes No

```
public class Solution {
    public ListNode rotateRight(ListNode A, int B) {
        if(A==null){
            return null;
        }
        int n=1;
        ListNode dummy=A;
```

```
        while(dummy.next!=null){
           dummy=dummy.next;
           n++;
        }
        ListNode temp=A;
        for(int i=n-B%n;i>1;i--){
           temp=temp.next;
        }
        dummy.next=A;
        A=temp.next;
        temp.next=null;
        return A;

    }
}
```

List Cycle

Asked in: Amazon Microsoft NetApp

Suggest Edit Bookmark

Given a linked list, return the node where the cycle begins. If there is no cycle, return `null`.

Try solving it using constant additional space.

Example :

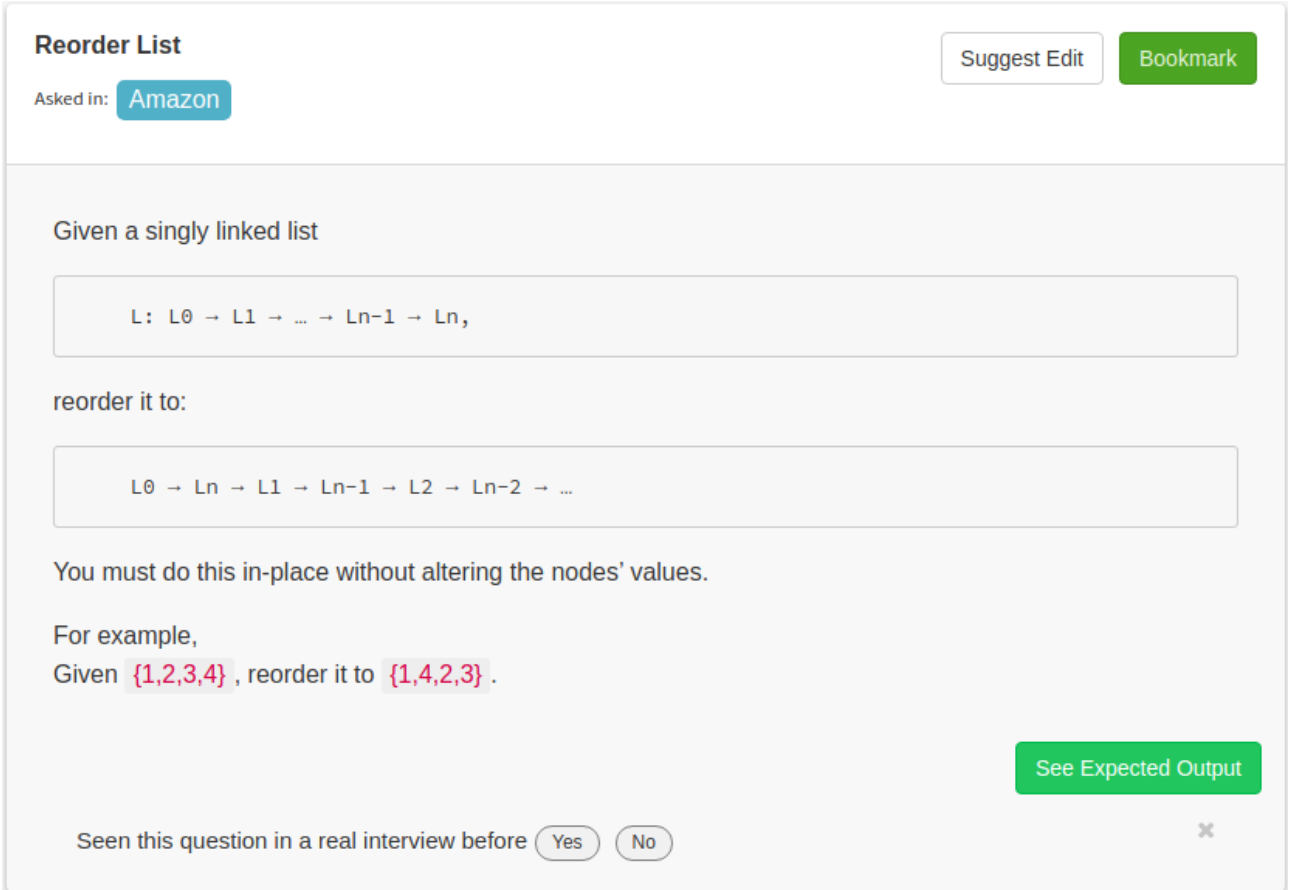
```
Input :

                ______
               |      |
               \/     |
        1 -> 2 -> 3 -> 4

Return the node corresponding to node 3.
```

See Expected Output

Seen this question in a real interview before Yes No

```
/**
 * Definition for singly-linked list.
 * class ListNode {
 *     public int val;
 *     public ListNode next;
 *     ListNode(int x) { val = x; next = null; }
 * }
 */
```

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



Reorder List

Asked in: Amazon

Suggest Edit | Bookmark

Given a singly linked list

```
L: L0 → L1 → … → Ln-1 → Ln,
```

reorder it to:

```
L0 → Ln → L1 → Ln-1 → L2 → Ln-2 → …
```

You must do this in-place without altering the nodes' values.

For example,
Given {1,2,3,4}, reorder it to {1,4,2,3}.

See Expected Output

Seen this question in a real interview before Yes No

```
/**
 * Definition for singly-linked list.
 * class ListNode {
 *     public int val;
 *     public ListNode next;
 *     ListNode(int x) { val = x; next = null; }
```

```
 * }
 */
 //I add the comment whcih help me help in revise
public class Solution {
    public ListNode reorderList(ListNode A) {
      if(A==null || A.next==null){
          return A;
      }
      ListNode slow=A;
      ListNode fast=A;
      ListNode prev=null;
      while(fast!=null && fast.next!=null){
          prev=slow;
          slow=slow.next;
          fast=fast.next.next;
      }
      prev.next=null;
      ListNode B=reverse(slow);
      merge(A,B);
      return A;
    }
    public static void merge(ListNode l1,ListNode l2)
    {
        while(l1!=l2)
        {
            ListNode n1=l1.next;
            ListNode n2=l2.next;
            l1.next=l2;
            if(n1==null){
                break;
            }
            l2.next=n1;
            l1=n1;
            l2=n2;
        }
    }

    public static ListNode reverse(ListNode A)
    {
        ListNode cur=A,prev=null,next=null;
        while(cur!=null)
        {
            next=cur.next;
            cur.next=prev;
            prev=cur;
            cur=next;
        }
        return prev;
    }


}
```

Partition List

Asked in: Microsoft

Suggest Edit | Bookmark

Given a linked list and a value x, partition it such that all nodes less than x come before nodes greater than or equal to x.

You should preserve the original relative order of the nodes in each of the two partitions.

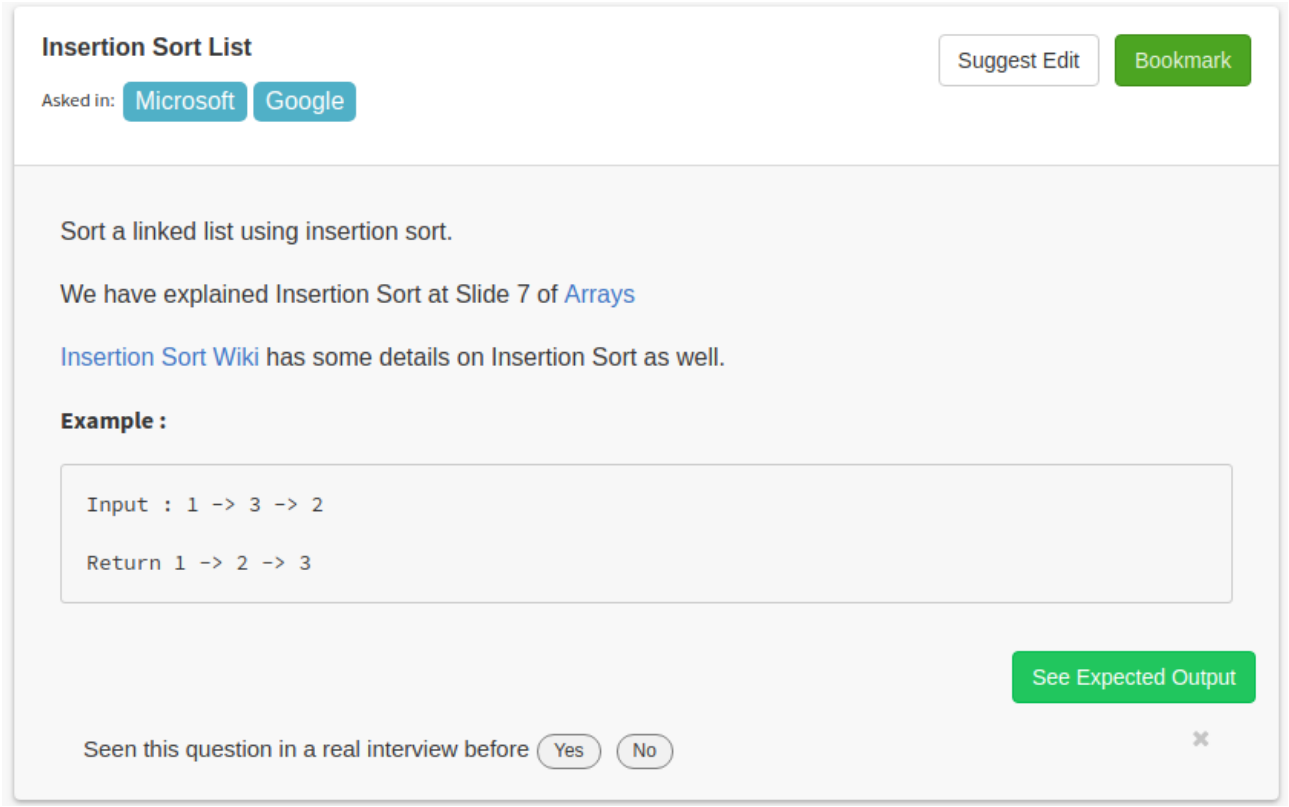
For example,
Given `1->4->3->2->5->2` and `x = 3`,
return `1->2->2->4->3->5`.

See Expected Output

Seen this question in a real interview before Yes No

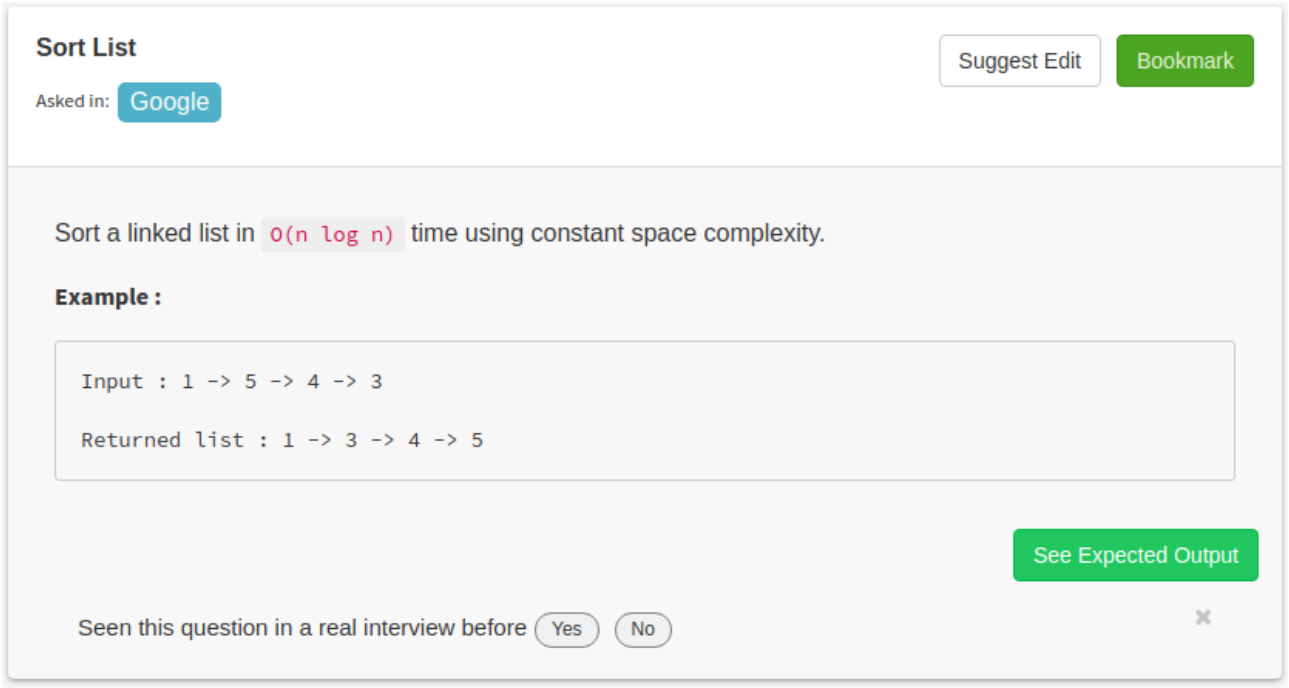
```
/**
 * Definition for singly-linked list.
 * class ListNode {
 *     public int val;
 *     public ListNode next;
 *     ListNode(int x) { val = x; next = null; }
 * }
 */
public class Solution {
    public ListNode partition(ListNode A, int B) {
        ListNode dummy1=new ListNode(0);
        ListNode dummy2=new ListNode(0);
        ListNode temp1=dummy1;
        ListNode temp2=dummy2;
        ListNode preserve1=temp1;
        ListNode preserve2=temp2;

        while(A!=null){
            if(A.val<B){
                temp1.next=A;
                temp1=A;
            }
            else{
                temp2.next=A;
                temp2=A;
            }
            A=A.next;
        }
        temp2.next=null;
        temp1.next=preserve2.next;
        return preserve1.next;
```

```
    }
}
```




```
/**
 * Definition for singly-linked list.
 * class ListNode {
 *     public int val;
 *     public ListNode next;
 *     ListNode(int x) { val = x; next = null; }
 * }
 */
public class Solution {
    public ListNode insertionSortList(ListNode A) {
        ListNode dummy=new ListNode(0);
        ListNode cur=A;
        ListNode next_node=dummy;
        ListNode next=null;
        while(cur!=null){
            next=cur.next;
            while(next_node.next!=null && next_node.next.val<cur.val){
                next_node=next_node.next;
            }
            cur.next=next_node.next;
            next_node.next=cur;
            next_node=dummy;
            cur=next;
        }
        return dummy.next;
```

```
    }
  }
```




```
/**
 * Definition for singly-linked list.
 * class ListNode {
 *     public int val;
 *     public ListNode next;
 *     ListNode(int x) { val = x; next = null; }
 * }
 */
public class Solution {
    public ListNode sortList(ListNode A) {
        if(A==null || A.next==null){
            return A;
        }
        ListNode slow=A;
        ListNode fast=A;
        ListNode stop=null;
        while(fast!=null && fast.next!=null){
            stop=slow;
            slow=slow.next;
            fast=fast.next.next;
        }
        stop.next=null;
        ListNode l1=sortList(A);
        ListNode l2=sortList(slow);
        ListNode l=Merge(l1,l2);
        return l;
    }
    public ListNode Merge(ListNode l1,ListNode l2){
        ListNode dummy=new ListNode(0);
```

```
        ListNode m=dummy;
        while(l1!=null && l2!=null){
            if(l1.val<=l2.val){
                m.next=l1;
                l1=l1.next;
            }
            else{
                m.next=l2;
                l2=l2.next;
            }
            m=m.next;
        }
        if(l1!=null){
            m.next=l1;
        }
        if(l2!=null){
            m.next=l2;
        }
        return dummy.next;


    }

}
```

Add Two Numbers as Lists

Asked in: Amazon Qualcomm Microsoft Facebook

Suggest Edit | Bookmark

You are given two linked lists representing two non-negative numbers. The digits are stored in **reverse order** and each of their nodes contain a single digit. Add the two numbers and return it as a linked list.

Input: `(2 -> 4 -> 3)` + `(5 -> 6 -> 4)`
Output: `7 -> 0 -> 8`

```
342 + 465 = 807
```

Make sure there are no trailing zeros in the output list
So, `7 -> 0 -> 8 -> 0` is not a valid response even though the value is still 807.

See Expected Output

Seen this question in a real interview before Yes No

```
/**
 * Definition for singly-linked list.
 * class ListNode {
 *     public int val;
 *     public ListNode next;
 *     ListNode(int x) { val = x; next = null; }
 * }
 */
 import java.util.*;
public class Solution {
    public ListNode addTwoNumbers(ListNode A, ListNode B) {
        ListNode dummy=new ListNode(0);
        ListNode step=dummy;
        int c=0;
        while(A!=null || B!=null || c!=0){
            if(A!=null){
                c+=A.val;
                A=A.next;
            }
            if(B!=null){
                c+=B.val;
                B=B.next;
            }
            ListNode temp=new ListNode(c%10);
            step.next=temp;
            step=step.next;
            c=c/10;
        }
        return dummy.next;
    }
}
```

Reverse Link List II

Asked in: Facebook Microsoft Amazon

Suggest Edit | Bookmark

Reverse a linked list from position m to n. Do it in-place and in one-pass.

For example:
Given `1->2->3->4->5->NULL`, `m = 2` and `n = 4`,

return `1->4->3->2->5->NULL`.

> ***Note:***
> *Given m, n satisfy the following condition:*
> `1 ≤ m ≤ n ≤ length of list`. ***Note 2:***
> *Usually the version often seen in the interviews is reversing the whole linked list which is obviously an easier version of this question.*

See Expected Output

Seen this question in a real interview before Yes No

```
/**
 * Definition for singly-linked list.
 * class ListNode {
 *     public int val;
 *     public ListNode next;
 *     ListNode(int x) { val = x; next = null; }
 * }
 */
public class Solution {
    public ListNode reverseBetween(ListNode A, int B, int C) {
        if(A==null || A.next==null){
            return A;
        }
        ListNode dummy=new ListNode(0);
        dummy.next=A;
        ListNode temp1=dummy;
        for(int i=0;i<B-1;i++){
                temp1=temp1.next;
        }
        ListNode first=temp1.next;
        ListNode second=temp1.next.next;
        for(int i=0;i<C-B;i++){
            first.next=second.next;
            second.next=temp1.next;
            temp1.next=second;
            second=first.next;
```

```
        }
        return dummy.next;
    }
}
```

Swap List Nodes in pairs

Asked in: Microsoft

Suggest Edit | Bookmark

Given a linked list, swap every two adjacent nodes and return its head.

For example,
Given `1->2->3->4`, you should return the list as `2->1->4->3`.

Your algorithm should use only constant space. You may not modify the values in the list, only nodes itself can be changed.

See Expected Output

Seen this question in a real interview before Yes No

```
public class Solution {
    public ListNode swapPairs(ListNode A) {
        ListNode dummy=new ListNode(0);
        dummy.next=A;
        while(A.next!=null && A.next.next!=null){
            int temp=A.val;
            A.val=A.next.val;
            A.next.val=temp;
            A=A.next.next;
        }
        if(A.next!=null){
            int temp=A.val;
            A.val=A.next.val;
            A.next.val=temp;
        }
         return dummy.next;

    }
}
```