Intersection of two Linked Lists:

```
class Solution {
public:

    typedef ListNode Node;
    // int countNodes(Node*head){
    //     int count=0;
    //     while(head!=NULL){
    //         count++;
    //         head=head->next;
    //     }
    //     return count;
    // }
    ListNode *getIntersectionNode(ListNode *headA, ListNode *headB) {
        Node*p1=headA;
        Node*p2=headB;

        while(p1!=p2){
            if(p1==NULL){
                p1=headB;
            }
            else{
                p1=p1->next;
            }
            if(p2==NULL){
                p2=headA;
            }
            else{
                p2=p2->next;
            }
        }
        return p1;
        // int len1=countNodes(headA);
        // int len2=countNodes(headB);

        // //step-2
        // int diff=abs(len1-len2);

        // Node*p1=headA;
        // Node*p2=headB;
        // if(len1>len2){
        //     while(diff!=0){
        //         p1=p1->next;
        //         diff--;
```

```
        //     }
        // }
        // else{
        //      while(diff!=0){
        //         p2=p2->next;
        //         diff--;
        //     }
        // }
        // while(p1!=p2){
        //     p1=p1->next;
        //     p2=p2->next;
        // }
        // return p1;
    }
};
```

Remove Nth Node from End of Linked List:

```
class Solution {
public:
    typedef ListNode Node;
    ListNode* removeNthFromEnd(ListNode* head, int n) {
        Node*dummy=new Node(-1);
        dummy->next=head;

        Node*slow=dummy;
        Node*fast=dummy;

        for(int i=1;i<=n;i++){
            fast=fast->next;
        }

        while(fast->next!=NULL){
            slow=slow->next;
            fast=fast->next;
        }
        Node*p=slow->next;
        slow->next=p->next;
        p->next=NULL;
        return dummy->next;;
    }
};
```

Add Two Numbers represented as Linked List:

```
class Solution {
public:

    typedef ListNode Node;
    ListNode* addTwoNumbers(ListNode* l1, ListNode* l2) {
        Node*p1=l1;
        Node*p2=l2;
        Node*dummy=new Node(-1);
        Node*ptr=dummy;

        int carry=0;
        while(p1!=NULL||p2!=NULL){
            int data1;
            if(p1==NULL){
                data1=0;
            }
            else{
                data1=p1->val;
            }

            int data2;
             if(p2==NULL){
                data2=0;
            }
            else{
                data2=p2->val;
            }
            int sum=carry+data1+data2;
            int digit=sum%10;
            carry=sum/10;

            ptr->next=new Node(digit);
            ptr=ptr->next;

            if(p1!=NULL){
                p1=p1->next;
            }
            if(p2!=NULL){
                p2=p2->next;
            }
        }
        if(carry>0){
            ptr->next=new Node(carry);
            ptr=ptr->next;
        return dummy->next;
```

```
    }
};
```