Q-1

```
class Solution {
public:
    ListNode *getIntersectionNode(ListNode *headA, ListNode *headB) {
        ListNode* tempa=headA;
        ListNode* tempb=headB;
        int a=0;
        while(tempa!=NULL){
            a++;
            tempa=tempa->next;

        }
        tempa=headA;
        //
        int b=0;
        while(tempb!=NULL){
            b++;
            tempb=tempb->next;

        }
        tempb=headB;

        if(a>b){
            while(a>b){
                tempa=tempa->next;
                a--;
            }
            while(tempa!=NULL && tempa!=tempb){
                tempa=tempa->next;
                tempb=tempb->next;
            }
            return tempa;
        }
        else{
            while(b>a){
                tempb=tempb->next;
                b--;
            }
            while(tempa!=NULL && tempa!=tempb){
                tempa=tempa->next;
                tempb=tempb->next;
            }
```

```
            return tempa;

        }
        return NULL;


    }
};

Q-2
ListNode* reverse(ListNode* head){
        ListNode* tempa=head;
        ListNode* tempb=head;
        ListNode* tempc=NULL;
        while(tempa!=NULL){
            tempa=tempa->next;
            tempb->next=tempc;
            tempc=tempb;
            tempb=tempa;


        }
        return tempc;



    }
    ListNode* reverseKGroup(ListNode* head, int k) {
        if(head==NULL || head->next==NULL) return head;
        ListNode* tempa=head;
       // ListNode* tempb=head; //1 2 3 4 5
        ListNode* tempc=head;
        ListNode* tempd=new ListNode(-1);
        ListNode* tempf=tempd;
        while(tempa!=NULL){
            int a=1;
            while(tempa->next!=NULL && a!=k){
                tempa=tempa->next;


                a++;
```

```
        }
        if(a!=k) break;
        ListNode* tempx=tempa->next;


        // tempb->next=NULL;
        tempa->next=NULL;
        ListNode* tempe=reverse(tempc);
        tempd->next=tempe;
        while(tempd!=NULL && tempd->next!=NULL){
            tempd=tempd->next;
        }
        // if(tempa!=NULL){
        tempd->next=tempx;

        tempa=tempx;
        tempc=tempx;

        // }



    }
    return tempf->next;
```

Q-3-

Q-4(right)

```
ListNode* rotateRight(ListNode* head, int k) {
    int size=0;
    ListNode* b=head;
    ListNode* h=head;
    ListNode* tail;
    while(b!=NULL){
        size++;
        if(b!=NULL){
            tail=b;
        }
        b=b->next;

    }
```

```
if(size==0) return NULL;
int s=k%size;
ListNode* temp=head;
for(int i=1;i<size-s;i++){
    temp=temp->next;
}

tail->next=head;
head=temp->next;
temp->next=NULL;
return head;
```

Q-5:

```
ListNode* addTwoNumbers(ListNode* l1, ListNode* l2) {
        ListNode dummy(0);   // Dummy node to simplify code
        ListNode* tail = &dummy;  // Pointer to last node in result list

        int carry = 0;

        // Traverse both lists until both are null and no carry remains
        while (l1 != nullptr || l2 != nullptr || carry != 0) {
            int sum = carry;

            if (l1 != nullptr) {
                sum += l1->val;
                l1 = l1->next;
            }

            if (l2 != nullptr) {
                sum += l2->val;
                l2 = l2->next;
            }

            carry = sum / 10;
            int digit = sum % 10;

            tail->next = new ListNode(digit);
            tail = tail->next;
```

```
    }

    return dummy.next;
```