Week 1 Striver SDE Sheet Sol

1. Reverse a LinkedList

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 * };
 */
class Solution {
public:
    ListNode* reverseList(ListNode* head) {
        ListNode *newHead = NULL;
        while (head != NULL) {
            ListNode *next = head->next;
            head->next = newHead;
            newHead = head;
            head = next;
        }
        return newHead;
    }
};
```

2. Find the middle of LinkedList

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next)
{}
 * };
 */
class Solution {
public:
    ListNode* middleNode(ListNode* head) {
        ListNode *slow = head, *fast = head;
        while (fast && fast->next)
            slow = slow->next, fast = fast->next->next;
        return slow;
    }
};
```

3. Add two numbers as LinkedList

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 * };
 */
class Solution {
public:
    ListNode* addTwoNumbers(ListNode* l1, ListNode* l2) {
        ListNode *dummy = new ListNode();
        ListNode *temp = dummy;
        int carry = 0;
        while( (l1 != NULL || l2 != NULL) || carry) {
            int sum = 0;
```

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

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

            sum += carry;
            carry = sum / 10;
            ListNode *node = new ListNode(sum % 10);
            temp -> next = node;
            temp = temp -> next;
        }
        return dummy -> next;
    }
};
```