Linked List
(Assignment Solutions)

Solution 1:

```
int getSize(ListNode* head) {
        int size=0;
        while(head != NULL) {
            head = head->next;
            size++;
        }

        return size;
    }

    ListNode *getIntersectionNode(ListNode *headA, ListNode *headB) {
        int m = getSize(headA);
        int n = getSize(headB);
        ListNode* t1 = headA;
        ListNode* t2 = headB;
        int diff = 0;

        if(m >= n) {
            diff = m-n;
            for(int i=0; i<diff; i++) {
                t1 = t1->next;
            }
        } else {
            diff = n-m;
            for(int i=0; i<diff; i++) {
                t2 = t2->next;
            }
        }

        while(t1 != NULL && t2 != NULL && t1 != t2) {
```

```
            t1 = t1->next;
            t2 = t2->next;
        }

        if(t1 == NULL) {
            return NULL;
        } else {
            return t1;
        }
    }
```


Solution 2:

```
ListNode* deleteNodes(ListNode* head, int m, int n) {
        auto pre = head;
        while (pre) {
            for (int i = 0; i < m - 1 && pre; ++i) {
                pre = pre->next;
            }
            if (!pre) {
                return head;
            }
            auto cur = pre;
            for (int i = 0; i < n && cur; ++i) {
                cur = cur->next;
            }
            pre->next = cur ? cur->next : nullptr;
            pre = pre->next;
        }
        return head;
}
```

Solution 3 :

```
int getSize(ListNode* head) {
        int size = 0;
        while(head != NULL) {
            size++;
            head = head->next;
        }

        return size;
    }

    ListNode* KthFromEnd(ListNode* head, int k) {
        ListNode* temp = head;
        int size = getSize(head);
        for(int i=0; i<size-k; i++) {
            temp = temp->next;
        }

        return temp;
    }

    ListNode* swapNodes(ListNode* head, int k) {
        ListNode* left = head;
        for(int i=0; i<k-1; i++) {
            left = left->next;
        }

        ListNode* right = KthFromEnd(head, k);

        int temp = left->val;
        left->val = right->val;
        right->val = temp;
        return head;
    }
```

Solution 4 :

```
ListNode* oddEvenList(ListNode* head) {
        //case of LL length = 0, 1, 2
        if(head == NULL|| head->next == NULL || head->next->next == NULL) {
            return head;
        }

        ListNode *evenStart = head->next;
        ListNode *odd = head;
        ListNode *even = head->next;

        while(odd->next && even->next){
            odd->next = even->next;
            even->next = odd->next->next;
            odd = odd->next;
            even = even->next;
        }
        odd->next = evenStart;
        return head;
}
```

Solution 5 :

```
ListNode* merge(ListNode* list1, ListNode* list2) {
        ListNode* merged = new ListNode(-1);
        ListNode* mptr = merged;
        ListNode* left = list1;
        ListNode* right = list2;

        while(left != NULL && right != NULL){
            if(left->val <= right->val){
                mptr->next = left;
                mptr = mptr->next;
```

```
            left = left->next;
        }
        else{
            mptr->next = right;
            mptr = mptr->next;
            right = right->next;
        }
    }

    if(left != NULL){
        mptr->next = left;
    }

    if(right != NULL){
        mptr->next = right;
    }

    return merged->next;
}


ListNode* mergeKLists(vector<ListNode*>& lists) {
    if(lists.size() == 0){
        return NULL;
    }
    if(lists.size() == 1){
        return lists[0];
    }
    ListNode* head = lists[0];
    for(int i=1; i<lists.size(); i++){
        head = merge(head, lists[i]);
    }

    return head;
}
```