Section #6

Tyler Conklin and Kate Rydberg (Based on handouts by previous CS106B instructors and TAs, especially those of Ashley Taylor and Shreya Shankar.)

Pointers and Linked Lists: Solutions

1. Linked Nodes Before/After

```
a) ListNode* temp = list->next->next;  // temp -> 3
   temp->next = list->next;            // 3 -> 4
   list->next->next = list;            // 4 -> 5
   list->next->next->next = nullptr;   // 5 /
   list = temp;
b) list->next->next->next = list;      // 3 -> 5
   list = list->next->next;            // list -> 3
   ListNode* list2 = list->next->next; // list2 -> 4
   list->next->next = nullptr;         // 5 /
```

2. Is Sorted

```
bool isSorted(ListNode* front) {
    if (front != nullptr) {
        ListNode* current = front;
        while (current->next != nullptr) {
            if (current->data > current->next->data) {
                return false;
            }
            current = current->next;
        }
    }
    return true;
}
```

3. Insert

```
void insert(ListNode*& front, int index, string value) {
    if (index == 0) {
        front = new ListNodeString(value, front);
    } else {
        ListNodeString* temp = front;
        for (int i = 0; i < index - 1; i++) {
            temp = temp->next;
        }
        temp->next = new ListNodeString(value, temp->next);
    }
}
```

4. Merge

```
ListNode* merge(ListNode* a, ListNode* b) {
    if (a == nullptr) {
        return b;
    }
    if (b == nullptr) {
        return a;
    }
    if (a->data <= b->data) {
        a->next = merge(a->next, b);
        return a;
    } else {
        b->next = merge(a, b->next);
        return b;
    }
}
```

5. Remove All

```
void removeAll(ListNode*& front, int value) {
    while (front != nullptr && front->data == value) {
        ListNode* trash = front;
        front = front->next;
        delete trash;
    }
    if (front != nullptr) {
        ListNode* current = front;
        while (current->next != nullptr) {
            if (current->next->data == value) {
                ListNode* trash = current->next;
                current->next = current->next->next;
                delete trash;
            } else {
                current = current->next;
            }
        }
    }
}
```

6. Split

```
void split(ListNode*& front) {
    if (front != nullptr) {
        ListNode* current = front;
        while (current->next != nullptr) {
            if (current->next->data < 0) {
                ListNode* temp = current->next;
                current->next = current->next->next;
                temp->next = front;
                front = temp;
            } else {
                current = current->next;
            }
        }
    }
}
```