CS 106B Section 9 (Week 10) Solutions

---

1. consume

```
void LinkedList::consume(LinkedList& other) {
    if (front == NULL) {
        front = other.front;
    } else {
        ListNode* current = front;
        while (current->next != NULL) {
            current = current->next;
        }
        current->next = other.front;
    }
    other.front = NULL;
}
```

---

2. transferEvens

```
LinkedList* LinkedList::transferEvens() {
    LinkedList* result = new LinkedList();
    if (front != NULL) {
        result->front = front;
        front = front->next;
        ListNode* current = front;
        ListNode* resultLast = result->front;
        while (current != NULL && current->next != NULL) {
            resultLast->next = current->next;
            resultLast = current->next;
            current->next = current->next->next;
            current = current->next;
        }
        resultLast->next = NULL;
    }
    return result;
}
```

CS 106B Section 9 (Week 10) Solutions

---

3. hanoi

```
void hanoi(int disks) {
    hanoi(disks, 1, 3);
}

void hanoi(int disks, int from, int to) {
    if (disks > 0) {
        int thirdPeg = 6 - from - to;
        hanoi(disks - 1, from, thirdPeg);
        cout << "move disk " << disks << " from peg " << from
                                      << " to peg " << to << endl;
        hanoi(disks - 1, thirdPeg, to);
    }
}
```

---

4. sequential

```
TreeNode* sequential(int n) {
    TreeNode* node = new TreeNode;
    sequentialHelper(node, 0, n - 1);
    return node;
}

void sequentialHelper(TreeNode*& node, int low, int high) {
    if (low > high) {
        node = NULL;
    } else {
        int mid = (low + high) / 2;
        node = new TreeNode(mid);
        sequentialHelper(node->left, low, mid - 1);
        sequentialHelper(node->right, mid + 1, high);
    }
}
```