CS 106B Section 9 (Week 10)

This week is an optional review for the final. If you do not get to all of them in section, make sure to do the rest on your own for practice!

---

1. consume *(LinkedLists).*

Write a member function named **consume** that could be added to the LinkedList class. It should accept a reference to another LinkedList and return void. After the call is done, all the elements from the second LinkedList should be appended to the end of the first one in the same order, and the second LinkedList should be empty. Note that either linked list can be empty. For example, if we start with

```
list1: {1,3,5,7}
list2: {2,4,6}
```

list1.consume(list2) produces	**list2.consume(list1)** produces
list1: {1,3,5,7,2,4,6}	list1: {}
list2: {}	list2: {2,4,6,1,3,5,7}

```
void LinkedList::consume(LinkedList& other)  { ...
```

---

2. transferEvens *(LinkedLists).*

Write a member function **transferEvens** that could be added to the LinkedList class. It should remove the even-index elements from its linked list and return a pointer to a new LinkedList object with those elements (in the same order). For example:

```
LinkedList* list1 : {3,1,4,15,9,2,6,5,35,89}
```

```
LinkedList* list2 = list1→transferEvens() produces

list1 : {1,15,2,5,89}
list2 : {3,4,9,6,35}
```

```
LinkedList* LinkedList::transferEvens() { ...
```

CS 106B Section 9 (Week 10)

3. hanoi *(recursion).*

The **Tower of Hanoi** is a game where you have three pegs (#1, #2, and #3) and some circular disks of different sizes that slide onto the pegs. They all start on one peg, largest to smallest (largest on the bottom). The goal is to move all the disks to another peg by following these rules: you may only move one disk at a time from peg to peg; no disk may be placed on top of a smaller disk. Play through a simple example to figure out how to win the game.

Write a function named **hanoi** that takes one int, the number of disks and returns void. The function should print the solution to the game where those disks start on peg 1 and must end up on peg 3. For example, hanoi(3) should print: (where disk 1 is the smallest and disk 3 is the largest)

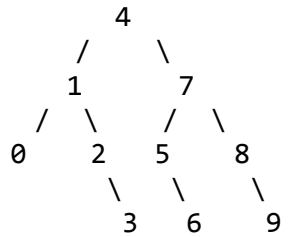
```
move disk 1 from peg 1 to peg 3
move disk 2 from peg 1 to peg 2
move disk 1 from peg 3 to peg 2
move disk 3 from peg 1 to peg 3
move disk 1 from peg 2 to peg 1
move disk 2 from peg 2 to peg 3
move disk 1 from peg 1 to peg 3
```

```
void hanoi(int disks) { ...
```

---

4. sequential (*binary trees).*

Write a function named **sequential** that takes a single int N and returns a pointer to a TreeNode that is the root of a tree with N nodes, with data fields 0 through N-1, such that the in-order traversal of the tree has data fields in increasing order, and the tree is balanced. (The number of nodes in *any* left subtree should be within one of the number of nodes in the corresponding right subtree.) If there are an odd number of nodes, put the extra node in the right subtree. For example, sequential(10) must create the following tree and return the pointer to the 4 node:




```
TreeNode* sequential(int n) { ...
```