CS 61B Discussion 2: Pointers Fall 2015

1 Boxes and Pointers

Draw a box and pointer diagram to represent the IntLists after each statement.

```
IntList L = IntList.list(1, 2, 3, 4);
IntList M = L.tail.tail;
N = IntList.list(5, 6, 7);
N.tail.tail.tail = N;
L.tail.tail = N.tail.tail.tail.tail;
M.tail.tail = L;
```

2 Shifting a Linked List

Implement the following methods to circularly shift an IntList to the left destructively and non-destructively.

```
/** Destructively shifts the elements of the given IntList L to
  * the left by one position (e.g. if the original list is
  * (5, 4, 9, 1, 2, 3) then this method should return the list
  * (4, 9, 1, 2, 3, 5)). Returns the first node in the shifted list.
  * Don’t use ’new’; modify the original IntList. */
public static IntList shiftListDestructive(IntList L) {









}
```

```
/** Non-destructively shifts the elements of the given IntList L
  * to the left by one position. Returns the first node in the shifted list.
  * Don't modify the original IntList. */
public static IntList shiftListNondestructive(IntList L) {









}
```

3 Palindrome

Implement the following two methods which determine whether an IntList is a palindrome.

```
/** Non-destructively reverses an IntList L.
  * Do not modify the original IntList. */
public static IntList reverseNondestructive(IntList L) {









}

/** Returns whether the IntList L is a palindrome or not,
  * or if it reads the same backwards as forwards. Hint: you may
  * want to use reverseNondestructive. */
public static boolean isPalindrome(IntList L) {









}
```