CS 61B Discussion 2 Spring 2015

1 Practice with Linked Lists

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

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

N.tail.tail = N;
M.tail.tail.tail = N.tail;
L.tail.tail = L.tail.tail.tail;
L = M.tail;
```

2 Squaring a List

Write the following methods to destructively and non-destructively square a linked list.

```
/** Destructively squares each element of the given IntList L.
 * Don't use 'new'; modify the original IntList.
 * Should be written iteratively. */
    public static IntList SquareDestructive(IntList L) {
       IntList B = L;
       while (B != null) {
           B.head *= B.head;
           B = B.tail
       }
       return L;







    }
```

```
/** Non-destructively squares each element of the given IntList L.
 * Don't modify the given IntList.
 * Should be written recursively.*/
     public static IntList SquareNonDestructive(IntList L) {
        if (L == null) {
                return L;
        }
        else {
                IntList tail = SquareNonDestructive(L.tail);
                IntList M = new IntList(L.head * L.head, tail);
                return M;
        }




}
```

Bonus for bosses: Write `SquareDestructive` recursively. Write `SquareNonDestructive` iteratively.

3 Reversing Linked Lists

```
/** Takes in an IntList and non-destructively returns an IntList whose
   elements have been reversed.*/
    public static IntList reverseNonDestructive(IntList lst) {
                IntList L2 = null;
                while (L != null) {
                        L2 = new IntList(L.head, L2);
                        L = L.tail;
                }
                return L2;


    }

    /** Bonus for bosses: Write reverseDestructive, which takes in an IntList
       and destructively returns the same IntList with reversed elements.
       You should not use 'new'.*/
    public static IntList reverseDestructive(IntList L) {
                if(L == null || L.tail == null) {
                        return L;
                } else {
                        IntList newHead = reverse(L.tail);
                        L.tail.tail = L;
                        L.tail = null;
                        return newHead;
                        }
                }
        }
```