

1 Practice with Linked Lists

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

```
1      IntList L = new IntList(4, null);
2      L = new IntList(3, L);
3      L = new IntList(2, L);
4      L = new IntList(1, L);
5      IntList M = L.tail;
6      IntList N = new IntList(6, null);
7      N = new IntList(5, N);
8
9      N.tail.tail = N;
10     M.tail.tail.tail = N.tail;
11     L.tail.tail = L.tail.tail.tail;
12     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) {

    }
}
```

```
/** 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) {
```

```
}
```

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) {
```

```
}
```

```
/** 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 void reverseDestructive(IntList L) {
```