CS 61B Discussion 2: Pointers Fall 2018

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

2 Reverse

Implement the following method, which reverses an IntList non-destructively.

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









}
```

Extra: Implement the following method which destructively reverses an IntList L

```
/** Destructively reverses an IntList L. */
public static IntList reverseDestructive(IntList L) {









}
```

3 Insertion

Implement the following method to insert an element into the given position of an IntList. This method should modify the list L and should not create a new list.

```
/** Insert a new item at the given position in L and return the resulting
  * IntList. If the position is past the end of the list, insert a new
  * node at the end of the list. For example if L is (1, 2, 4) then the
  * result of insert(L, 3, 2) would be (1, 2, 3, 4) */
public static IntList insert(IntList L, int item, int position) {









}
```

4 *Extra:* Shifting a Linked List

Implement the following methods to circularly shift an IntList to the left 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) {









}
```