CS 61B
Spring 2019

Extra Practice Worksheet
Discussion 2: Pointers

Worksheet 2

1 Pointer Practice (Fa17 Tutor Worksheet)

Draw the resulting box and pointer diagram for the L1 Singly Linked IntList after the following code is executed:

1. **IntLists**

```
IntList L1 = IntList.list(2,4,6,8);
IntList L2 = IntList.list(1,3,5,7);
L1.tail.tail.head = 5;
L2.tail.tail.tail = L1;
L1.tail.tail.tail = L2;
```

2. **IntLists**

```
IntList L1 = IntList.list(7,15,22,31);
IntList L2 = L1.tail.tail;
L2.tail.head = 13;
L1.tail.tail.tail = L2;
IntList L3 = IntList.list(50);
L2.tail.tail = L3;
```

2 Destructivity

Will is working on his app, CalTransit (check it out on the Apple App Store!) and is writing a function that given an Intlist, appends the length of the Intlist at the end of the list. Tiger thinks writing a non destructive function will be a better idea. Tiger writes the following method:

```
public static IntList addLength(IntList i):
    temp = i
    temp.addLast(i.length)
    return temp
```

Assuming that the IntList class was already correctly defined, will this method execute as expected? If not, how can it be fixed?

3 Skipping Stones (Fa17 Tutor Worksheet)

Write a function that takes in an IntList *L*, which must contain at least one element, and returns an IntList with every odd indexed element removed. Try out both the destructive and nondestructive approaches.

1. **Nondestructive**

```
public static IntList skipNondestructive (IntList L) {
    IntList pointer = ____________________________;
    IntList _________= ____________________________;
    while (____________________________&& ____________________________) {
        L = ____________________________________________;
        pointer.tail = ____________________________________;
        pointer = ____________________________________________;
    }
return ____________________________________________;     }
```

2. **Destructive**

```
public static IntList skipDestructive (IntList L) {
    IntList pointer = ____________________________________;
    while (____________________&& ____________________) {
```

```
            _____;
            _____;
        }
        _____
        return _____;
    }
```