CS 61B Spring 2020 | Small Group Tutoring Section 1: Pointers | Worksheet 3

1 Java Practice

1. Write a function that sums up all the digits in an integer iteratively. For example, sumDigits(31415) should return $3+1+4+1+5=14$.

```
public static int sumDigits (int x) {




}
```

2. Write a function that sums up all the digits in an integer recursively.

```
public static int sumDigits (int x) {
    if (_____) {
        _____;
    }
    return _____+ sumDigits(_____);
}
```

2 Pointer Practice

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;
```

3 Skip Me

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, starting at index 0. For example, if $L = \{1,2,3,4\}$, the function should return an IntList with elements $\{1,3\}$.

1. **Nondestructive**: input IntList, L, should not be modified

```
public static IntList skipNondestructive (IntList L) {
    IntList pointer = _____________________________;
    IntList retPtr = _____________________________;
    IntList retHead = _____________________________;
    while (_____________________________&& _____________________________) {
        retPtr.tail = _____________________________________________________;
        pointer = _____________________________________________;
        retPtr = _____________________________________________;
    }
    return _____________________________________________;
}
```

2. **Destructive**: input IntList, L, should be modified

```
public static void skipDestructive (IntList L) {
    if (_____________________________________________) {
        _____________________________________________;
    }
    L.tail = _____________________________________________;
    skipDestructive(_____________________________);
}
```