

## 1 Pointer Practice

Draw the resulting box and pointer diagram for the `IntLists` after the following code is executed:

### **IntLists**

```
IntList L1 = IntList.of(7, 15, 22, 31);  
IntList L2 = L1.next.next;  
L2.next.item = 13;  
L1.next.next.next = L2;  
IntList L3 = new IntList(50);  
L2.next.next = L3;
```

## 2 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. **Destructive:** `IntList L` should be modified

```
public static void skipDestructive (IntList L) {
    if ( _____ ) {
        _____;
    }
    L.next = _____;
    skipDestructive( _____ );
}
```

### 2. **Nondestructive:** `IntList L` should not be modified

```
public static IntList skipNondestructive (IntList L) {
    IntList pointer = _____;
    IntList retPtr = _____;
    IntList retHead = _____;
    while ( _____ && _____ ) {
        retPtr.next = _____;
        pointer = _____;
        retPtr = _____;
    }
    return _____;
}
```

## 3 Benefits of Enhancements

1. List one advantage of having a sentinel node.
2. Suppose we implement a doubly linked list with a sentinel. In order to write the `addFirst` method, which pointers will we change?

```
sentinel.next
sentinel.prev
sentinel.next.prev
sentinel.next.next
```