Lab 5 Notes

Sunday, June 30, 2019 8:10 PM

CS 61BL
Summer 2019

Lab 5
July 1, 2019

Name:

SID:

Section Number: 01 [✓ 02] 03 04 05 06 07 08 09 10 11 12

Write your name and login above. Please complete this worksheet during your lab, and turn it in to your TA by the end of your section. You are encouraged to work with your partners and neighbors collaboratively.

1 IntList Box and Pointer Diagram

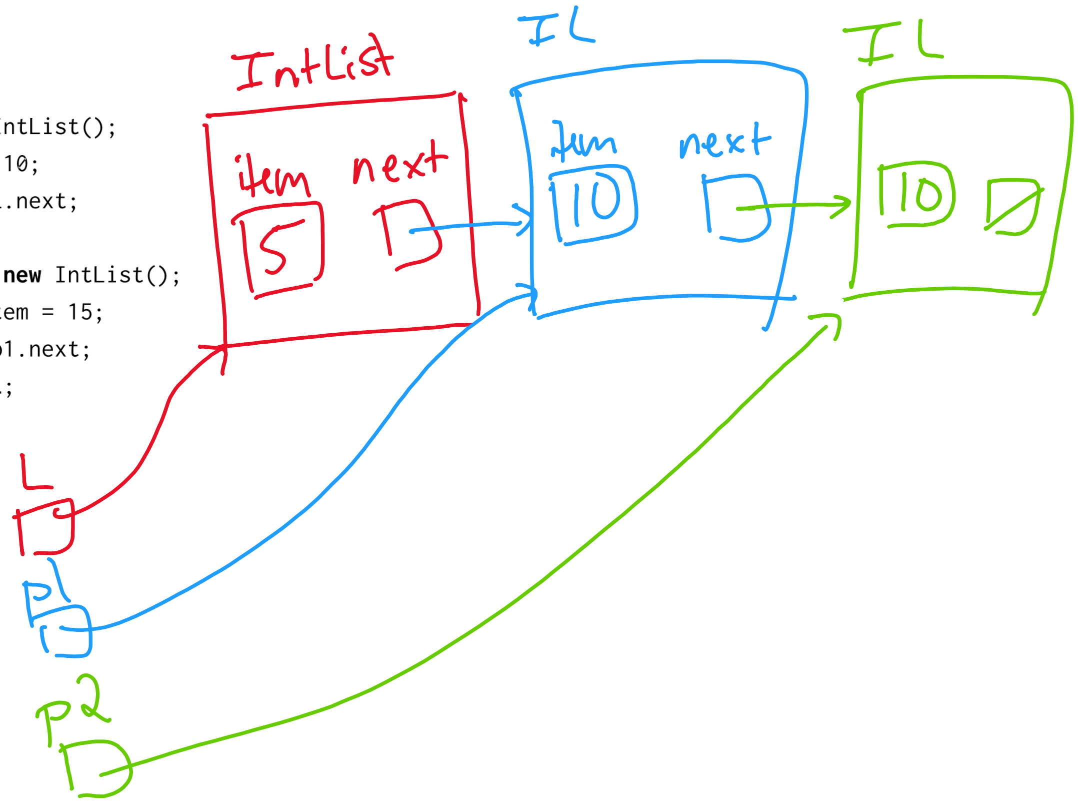
1.1 Draw the box-and-pointer diagram that results from running the following code.

```
public class IntList {
    public int item;
    public IntList next;

    public static void main(String[] args) {
        IntList L = new IntList();
        L.item = 5;
        L.next = null;

        L.next = new IntList();
        L.next.item = 10;
        IntList p1 = L.next;

        L.next.next = new IntList();
        L.next.next.item = 15;
        IntList p2 = p1.next;
        p1.next = null;
    }
}
```



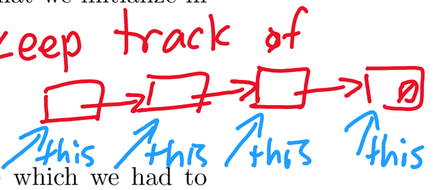
2 Iteration vs. Recursion

2.1 Consider the two different implementations of the `size` method provided in the lab spec. The method is first implemented using iteration (with a `while` loop) and the other method is implemented using recursion.

Answer the following questions with your partner and write down your conclusions.

(a) How does the recursive `size` keep track of the pointer, `p`, that we initialize in `iterativeSize`?

recursive call, has this to keep track of current element.

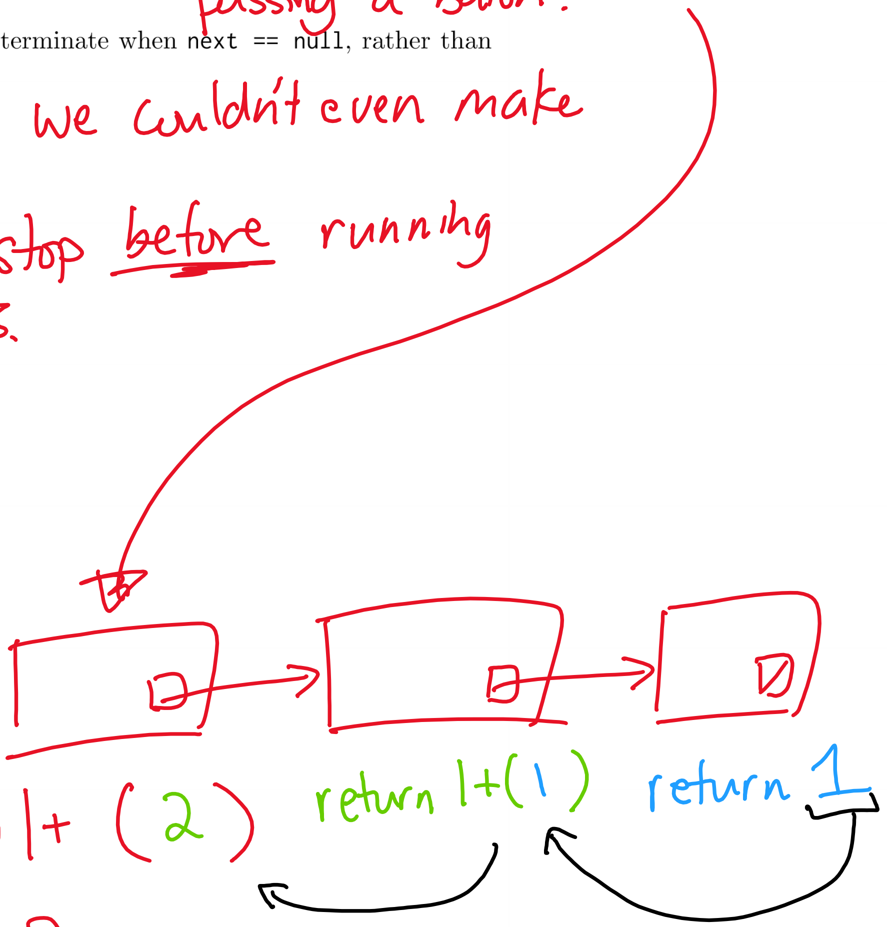


(b) How does the recursive `size` keep track of the `totalSize` which we had to create a variable for in `iterativeSize`?

does not have a single variable. Relies on recursive Calls returning with the correct answer. Kind of like passing a baton.

(c) Why does the base case for `size` terminate when `next == null`, rather than when `this == null`?

if this == null we couldn't even make this call!
We want to stop before running out of elements.



another analogy, we have a line and you want to find what place you are, you ask "what place in line are you?" to the person ahead of you. they do the same, until we reach the front, where the first person says "1"
⇒ 2nd knows they are 2, 3rd knows they are 3, ...

3