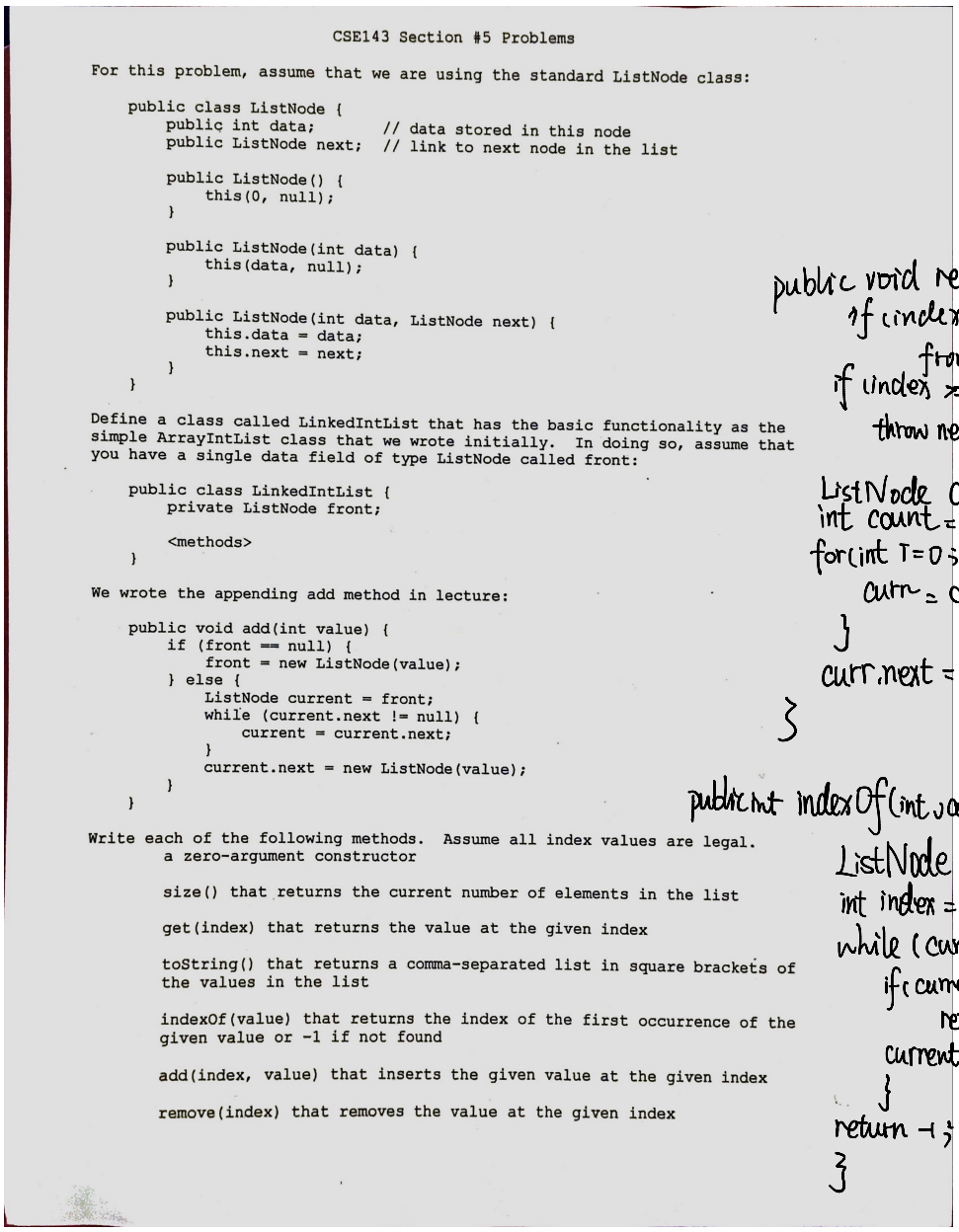
Quiz

Thursday, October 11, 2018 3:30 PM



CSE143 Section #5 Problems

For this problem, assume that we are using the standard ListNode class:

```
public class ListNode {
    public int data;       // data stored in this node
    public ListNode next;  // link to next node in the list

    public ListNode() {
        this(0, null);
    }

    public ListNode(int data) {
        this(data, null);
    }

    public ListNode(int data, ListNode next) {
        this.data = data;
        this.next = next;
    }
}
```

Define a class called LinkedIntList that has the basic functionality as the simple ArrayIntList class that we wrote initially. In doing so, assume that you have a single data field of type ListNode called front:

```
public class LinkedIntList {
    private ListNode front;

    <methods>
}
```

We wrote the appending add method in lecture:

```
public void add(int value) {
    if (front == null) {
        front = new ListNode(value);
    } else {
        ListNode current = front;
        while (current.next != null) {
            current = current.next;
        }
        current.next = new ListNode(value);
    }
}
```

Write each of the following methods. Assume all index values are legal.
a zero-argument constructor

size() that returns the current number of elements in the list

get(index) that returns the value at the given index

toString() that returns a comma-separated list in square brackets of the values in the list

indexOf(value) that returns the index of the first occurrence of the given value or -1 if not found

add(index, value) that inserts the given value at the given index

remove(index) that removes the value at the given index

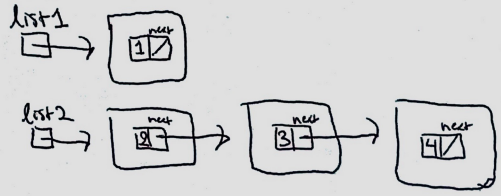
```
public void remove(index) {
    if (index == 0)
        front = null;
    if (index >= size() || index < 0)
        throw new IllegalArgumentException;
    ListNode curr = front;
    int count = 0;
    for(int i=0; i<index-1; i++) {
        curr = curr.next;
    }
    curr.next = curr.next.next;
}
```

```
public int indexOf(int value) {
    ListNode current = front;
    int index = 0;
    while (current != null) {
        if (current.data == value)
            return index;
        current = current.next;
    }
    return -1;
}
```

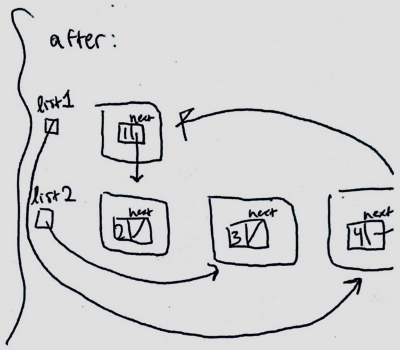
```
public int deleteBack() {
    if(front == null || front.next == null){
        return null;
    }
    ListNode cur = front;
    while(cur.next.next != null) {
        cur = cur.next;
    }

    // found the last ListNode and stopped
    int dataBack = cur.data;
    cur.next = null;
    return dataBack;
}
```

[redacted]: write code to rearrange ListNodes to get from the before pictures to the after pictures

before:



after:



l2: 3

l1: 4 1 2

```
list2.next.next.next = list1;
list1 = list2;
list1.next = list2.next.next;
list2.next.next = null;
list1.next.next = list2;
list 2 = list1.next.next.next;
```