Name: _____

CS2113 Quiz 1

45 minutes (you may find the bug in a fraction of this time).

If you are unsure of what any line of code below does, you may ask your TA – they cannot help you debug or trace memory, but they can explain a line of code to you in English.

The code below is a class that has a method with **TWO BUGS** that cause at least two of the assertions below to fail.

Trace the code, like we did in class last week for the Dictionary problem, and **CIRCLE** the line with the bug.

Then, fix that line so that it passes all the test cases, **changing only one line of code for each bug** (i.e. do not rewrite the entire method, just fix the broken line). Hint: remember the `substring(startIndex, endIndex)` method of the `String` class.

We will apply the following grading rubric:

- 25 points: code trace is correctly showing how memory changes over at least one test case input. You **must show a complete memory trace**, even if you know where the bug is, to receive credit here.
- 50 points: circled the correct TWO lines of code that contain the bugs that is causing the test case to fail
- 25 points: code is correctly fixed such that all tests pass

```
class PersonNode {
    String name;
    int age;
    PersonNode next;

    public PersonNode(String name, int age) {
        this.name = name;
        this.age = age;
        this.next = null;
    }

    public String getName(){
       return name;
    }

    public int getAge(){
       return age;
    }

    public PersonNode getNext(){
       return next;
    }
}

public class LinkedList {
    PersonNode head;

    public boolean search(String name, int age){
        PersonNode curr = head;
        boolean found = true;
        while(curr != null){
            if (curr.getName().equals(name) && curr.getAge() == age)
                return found;
            found = false;
            curr = curr.getNext();
        }
        return found;
    }

    public void add(PersonNode node) {
        PersonNode newNode = node;
        if (head == null) {
            head = newNode;
        } else {
            PersonNode temp = head;
            while (temp.next != null) {
                temp = temp.next;
            }
            temp.next = newNode;
        }
    }

    public String toString(){
        String result = "";
        PersonNode curr = head;
        while (curr != null){
            result += curr.getName() + " " + curr.getAge() + ", ";
            curr = curr.getNext();
        }
        return result;
    }
}
```

```
import org.junit.Test;
import static org.junit.Assert.assertEquals;


public class PersonTester {
	@Test
	public void test1(){
		LinkedList list = new LinkedList();
	 list.add(new PersonNode("Zoe", 11));
	 list.add(new PersonNode("Kim", 12));
	 list.add(new PersonNode("Ben", 13));
	 list.add(new PersonNode("Ben", 10));
	 list.add(new PersonNode("Joe", 14));
	 list.add(new PersonNode("Sam", 15));

	 String expected = "Zoe 11, Kim 12, Ben 13, Ben 10, Joe 14, Sam 15";
	 assertEquals(expected, list.toString());

	 assertEquals(true, list.search("Ben", 13));
	 assertEquals(true, list.search("Ben", 10));
	 assertEquals(false, list.search("Ben", 11));
	 assertEquals(false, list.search("Lee", 13));
	}
}
```

SCRATCH PAPER

SCRATCH PAPER