Name: ______________________________ Andrew Id: ____________________

15-121 Fall 2021 Quiz 8

Up to 20 minutes. Show your work. No calculators, no notes, no books, no computers, no other people.

1. (8 points) **Free Response**: Interfaces

   Consider a new abstract data type called a `Set`. A set is an unordered collection of unique items. A set allows you to do the following:

   - Add an item to the set (`add`)
   - Check if an item is in the set (`contains`)
   - Remove a specified item from the set (`remove`)
   - Merge another set into the current set (`union`)

   Write an interface for `Set`. You should use the method names listed in parenthesis above, but you need to choose the correct arguments and return type of each method. If you aren't sure about the arguments or return type for a method, then think about how that same method works for ADTs we have already studied.

   Note that you are only writing an interface.

2. **Free Response**: How many linked list questions can he possibly write?

   Assume the existence of the following singly linked list. (It is a simplified version of what was in the homework.) You can assume that `head` and `size` are all properly updated in any of the provided methods below.

```
public class SingleListNode<NodeDataType> {
    public NodeDataType data;
    public SingleListNode<NodeDataType> next;

    public SingleListNode(NodeDataType data) {
        this.data = data;
        this.next = null;
    }
}

public class SinglyLinkedList<ListType> {
    private SingleListNode<ListType> head;
    private int size;

    public SinglyLinkedList() {
        // You may assume this code exists and works as described in the homework.
    }

    public void add(ListType value) {
        // You may assume this code exists and works as described in the homework.
    }

    private boolean recursiveContainsHelper(SingleListNode<ListType> node, ListType value) {
        // You will write this code in part a.
    }

    public boolean contains(ListType value) {
        return recursiveContainsHelper(this.head, value);
    }

    public int count(ListType value) {
        // You will write this code in part b
    }

}
```

   (a) (6 points) Write the code for the **recursive** method `recursiveContainsHelper` that returns `true` if `value` can be found in the list that starts with `node`, and false otherwise. Your function *must* be recursive, and you may not use any loops.

```
private boolean recursiveContainsHelper(SingleListNode<ListType> node, ListType value) {
```

(b) (6 points) Write the code for the method `count` that returns the number of instances of `value` found in the list. If value is not in the list, then raise a `NoSuchElementException`.
This method does not need to be recursive, so feel free to use loops.

```
public int count(ListType value) {
```