CMSC 132 Quiz 2 Worksheet

The next quiz for the course will be on Tue, Jul 1. The following list provides more information about the quiz:

- The quiz will be a written quiz (no computer).
- The quiz will be in lab/discussion session.
- Closed book, closed notes quiz.
- Answers must be neat and legible. **You must use pencil.**
- Check the information available at http://www.cs.umd.edu/~nelson/classes/utilities/examRules.html

The following exercises cover the material to be included in this quiz. Solutions to these exercises will not be provided, but you are welcome to discuss your solutions with the TA or instructor during office hours. **We strongly recommend you do not use Eclipse to write the code associated with these exercises.** Try to answer the exercises in a piece of paper and then use Eclipse to verify your solutions. This approach will better prepare you for the quiz. **You cannot use any Java API class (except String) during the implementation of the methods below**.

Exercises

Implement the methods below based on the following Java class definitions.

```
public class LinkedList<T extends Comparable<T>> {
	private class Node {
		private T data;
		private Node next;

		private Node(T data) {
			this.data = data;
			next = null;
		}
	}

	private Node head;

	public LinkedList() {
		head = null;
	}

}
```

1. Define a method named **size** that returns the number of elements in the list.

2. Implement a method named **toString()** that returns a String with the data component of each node in the list.

3. Implement a method called **getCount** which has the following prototype:

```
public int getCount(T target)
```

   The method returns the number of instances of **target** in the list. Use the compareTo method for any comparisons.

4. Implement a method called **append** which has the following prototype:

```
public void append(ArrayList<T> data)
```

   The method appends all the elements from the ArrayList to the end of the LinkedList object. You must handle the case when the list is empty.

5. Implement a method called **removeLastInstance** that has the following prototype:

```
public void removeLastInstance(T value)
```

   The method removes the last instance of the parameter **value** present in the list.

6. Implement a method called **delete** that has the following prototype:

```
public LinkedList<T> delete(T target);
```

   The method will return a **new list** with all instances of **target** from the current object list removed. The original list should not be modified.