CMSC 132 Quiz 4 Worksheet

The fourth quiz for the course will be on Wednesday, March 14. The following list provides more information about the quiz:

- The quiz will be a written quiz (no computer).
- 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.

Some recursion problems require an auxiliary method. For example, a recursive implementation for the size() method of a linked list calls for an auxiliary method that takes as parameter a reference to a node. Keep this in mind when solving the problems below.

Exercises

Implement the methods below based on the following Java class definitions. For the following problems:

1. You must use recursion.
2. You may not modify the Node class
3. You may not add any instance variables to the LinkedList class. Adding auxiliary methods is OK.
4. You may not add any static variables to the LinkedList class.

```
public class LinkedList<T extends Comparable<T>> {
        public class Node {
                private T data;
                private Node next;
                public Node(T data) {
                        this.data = data;
                        next = null;
                }
        }
        Node head;
}
```

1. Define a **recursive** method that computes the size of a list.

2. Define a **recursive** method that prints the list in reverse order.

3. Define a **recursive** method ***boolean find(E elem)*** that returns true if **elem** is in the list and false otherwise.

4. Define a **recursive** method that returns the minimum value in the list. Keep in mind that the list is not sorted. Elements in the list must implement the Comparable interface.

5. Define a **recursive** method **boolean equals(LinkedList<E> list)** that determines whether two lists have the same corresponding elements. For example, the lists 10, 20, 30 and 10, 20, 30 are considered equals, whereas 10, 20, 30 and 30, 20, 10 are different.

6. Define a **recursive** method called **getCount** which has the following prototype:

   public int getCount(T targetElement)

   The method returns the number of instances of targetElement in the list.

7. Implement a **recursive** method named **removeLastElement** that removes the last element from the list.

8. Define a **recursive** method called **delete** that has the following prototype:

   public boolean delete(T target);

The method will delete the first instance of **target** from the list. The method will return true if a **target** instance is found and false otherwise. The list should not be modified if a **target** instance is not found.