CMSC 132 Quiz 3 Worksheet

The third quiz for the course will be on Wed, July 5. The following list provides additional information about the quiz:

- The quiz will be a written quiz (no computer).
- **The quiz will be in person in IRB – Antonov Room.**
- Closed book, closed notes quiz.
- Answers must be neat and legible.
- The quiz has a maximum duration of 20 minutes.
- Please do not discuss a quiz after you have taken it.
- Piazza will be closed on days quizzes take place.
- **Regarding Piazza -** Feel free to post questions in Piazza regarding the worksheet and possible solutions to problems.

The following exercises gives you practice with concepts that may show up on the quiz. Solutions to these exercises will not be provided, but you are welcome to discuss your solutions with the TAs during office hours. It is recommended that you try these exercises on paper first (without using a computer).

Exercises

1. Implement the methods below based on the following Java class definitions. For recursive methods you may only add one auxiliary function, and you may not add any instance nor static variables.

```
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;
   }
}
```

   a. Define a constructor that takes a **TreeSet<T>** as a parameter and initializes a linked list with the elements in the set. The new list must be sorted in increasing lexicographic order.

   b. Define a **RECURSIVE** method named **size** that returns the number of elements in the list. The prototype for this method is:

```
public int size()
```

   c. Define a **RECURSIVE** method named **inRange** that returns a **HashSet<T>** with the elements in the list that in the specified range. The range includes the lower and upper bound. The prototype for this method is:

```
public HashSet<T> inRange(T lowerBound, T upperBound)
```

   d. Define a **RECURSIVE** method named **remove** that removes all instances in the list that are equal to the target parameter. The prototype for this method is:

```
public void remove(T target)
```

e. Define a **RECURSIVE** method named **positionOfElementInList** that returns a **TreeMap<T, Integer>** that maps each element of the list to its position in the list. The prototype for this method is:

```
public TreeMap<T, Integer> positionOfElementInList()
```

f. Define an equals method for the class. Two lists are equal if the data elements in the two lists are the same (and appear in the same position).

2. Below are actual quizzes from past terms. I am just making this available for further practice, but remember your quiz in Summer 23 will be different than what we did in the past.

    a. http://www.cs.umd.edu/class/summer2023/cmsc132/quizzes/OldQ3.zip