**CMSC132 Fall 2018 Hashing, Lists, Sets Worksheet**

1. Is a hashCode method that returns 0 valid? Discuss.
2. Describe the data structure (e.g., classes) you will need to implement open addressing with linear probing.
3. What is the relationship that exists between a search key, a hash code, and a hash index?
4. Can a valid hashCode method return a negative value? Discuss.
5. 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;**

**}**

**public Set<T> removeInRange(boolean ordered, T lowerBound, T upperBound) {**

**// YOU MUST IMPLEMENT**

**}**

**private Node removeInRangeAux(Node headAux, T lowerBound, T upperBound, Set<T> newSet) {**

**// YOU MUST IMPLEMENT**

**}**

**}**

Implement the methods **removeInRange** and **removeInRangeAux** that will remove elements from the list that are in the range defined by **lowerBound** and **upperBound**. The elements that have been removed (if any) will be placed in a set. If the **ordered** parameter is true, the returned set will allow us to access the values in the order they were added to the set; otherwise the most efficient set type will be returned. To satisfy the recursive requirement, **removeInRange** calls the method **removeInRangeAux** (head = removeInRangeAux(…) will appear in **removeInRange**).