**CSE 274**

**Homework #5**

**Hash Set With Separate Chaining**

Except where noted, do not use any standard Java collection classes (e.g., TreeSet, TreeMap, HashSet, etc.) for this work.  
  
Submit a zipped up version of your Eclipse project. Insert your report (PDF file) in the top level directory of your source code.

1. **(5 pts)** Create a class to represent a car - Car - that year (int), make (String), model (String), and color (String). Car should implement Comparable and also override the equals and hashCode methods.
2. **(80 pts)** Create two class that implement a set using a hash table. Assume that the type of information being stored is Comparable and has proper equals and hashCode methods defined. This class will be similar to the HashedDictionary class presented in the textbook but will use separate chaining to resolve collisions. Another difference is that your class will not be a dictionary (i.e., key-value pairs), instead it will be used to represent a set. The individual buckets will be implemented in the following ways (only one approach will be active at any particular time):
   1. **(65 pts)** HashedSetSCJLL -- Separate chaining with Java’s LinkedList class. SCJLL represents “Separate Chaining using Java’s Linked List”
   2. **(15 pts)** HashedSetSCSLL - Use your own (i.e., or an adaptation of the textbook’s) sorted linked list to represent the buckets. SCSLL represents “Separate Chaining with Sorted Linked List”
3. **(15 pts)** Consider the two different implementations. In your report, identify at least one situation that you think each implementation will have the best performance, if this is possible. For example, you may think that class implementation XYZ will perform best when the load factor is high, the insertions occur at the beginning, followed by numerous calls to contains. The factors you might consider include: e.g., different table sizes/load factors, many insertions of a small range of numbers, many insertions of a wide range of numbers, many insertions followed by many calls to contains, etc. You are encouraged to run experiments to confirm/refute your conjectures.

Notes:

* Your hashed set implementations must implement the SetInterface given in the HW2 starter code. Remove the getCapacity() method.
* Your software should be highly polished. It should employ inheritance, interfaces, generics, helper methods, helper classes, etc. when appropriate. Be careful to reuse code when possible.
* Allocating and using the hashTable array will differ from the textbook’s examples. This is because of Java’s peculiar behavior with arrays of generic structures. The rough outline of the class will be:  
    
  class HashedSetSCJLL ... {  
   private LinkedList<T> [] hashTable;  
   public HashedSet() {  
   hashTable = (LinkedList<T>[])new LinkedList<?>[100];  
   }  
  }