

1. PROGRAM 6 INSTRUCTIONS, DATA STRUCTURES,  
PROF. LOFTIN, DUE MONDAY, MAY 3, 2010

Implement, inside a linked list class, a mergesort algorithm. Your linked list class should also implement Java generics.

You'll need to use in your directory the following files, for implementing Java generics. Node.java:

```
public class Node<E>{
    private E item;
    private Node<E> next;
    public Node(E newItem){
        item = newItem;
        next = null;
    }
    public Node(E newItem, Node<E> nextNode){
        item = newItem;
        next = nextNode;
    }
    public void setItem (E newItem){
        item = newItem;
    }
    public E getItem(){
        return item;
    }
    public void setNext (Node<E> nextNode){
        next = nextNode;
    }
    public Node<E> getNext(){
        return next;
    }
}
```

and ListInterface.java:

```
public interface ListInterface<E>{
    public boolean isEmpty ();
    public int size ();
    public void add (int index, E item);
    public E get (int index);
    public void remove (int index);
    public void removeAll();
}
```

For your code, you should convert the file `ListReferenceBased.java` to `ListWithMerge.java` by implementing Java generics and by including the following additional methods:

```
public void mergesort()
    // performs mergesort on the list object this
private void merge(ListWithMerge<E> secondList)
    // merges the sorted lists this and secondList into the list this
private ListWithMerge<E> splitList()
    // splits the list this into 2 halves, the first half to
    // remain in this, and the second half as the return value
```

The algorithm for mergesort is the same as in the array case:

- if in the base case (of  $\leq 1$  elements), return
- split the list into two equal pieces
- mergesort(the first half of the list)
- mergesort(the second half of the list)
- merge the two sorted halves.

Since the `mergesort` code uses the `Comparable` interface, we must insure that the Java generic class `E` implements `Comparable`. The syntax for the first line of the class should then be (see section 9.4 for details)

```
public class ListWithMerge <E extends Comparable<? super E>>
    implements ListInterface<E>{
```

## 2. HOW TO DO IT

Unlike the array implementation of mergesort, there is no need for an auxiliary temporary storage structure. Everything can be done by reassigning the `next` references of the `Nodes` (this is why it is useful to implement the `mergesort` method as a method of the list class: so the private data of the list can be accessed).

Make sure you update the `numItems` of the lists through the merging and splitting processes.

In the `merge` method, make sure you assign the correct version of the `head` reference.

## 3. DRIVER CLASS

Your program should work with the following driver class:

```
public class Driver{
    public static void main (String[] args){
        ListWithMerge<Integer> list = new ListWithMerge<Integer>();
        for (int i=0; i<20; i++)
```

```
        list.add(0, (int)(Math.random() * 100));
    printList (list);
    list.mergesort();
    printList (list);
}

public static void printList(ListInterface<Integer> list){
    for (int i=0; i<list.size(); i++)
        System.out.print (list.get(i) + " ");
    System.out.println();
}
}
```

The first line of the output should consist of a list of 20 random integers from 0 to 99. The second line should have the same list sorted in ascending order.

#### 4. DUE DATE

Monday, May 3, 2010.

#### 5. HOW TO TURN IT IN

Email your file `ListWithMerge.java` to `loftin@rutgers.edu` as an attachment. Be sure to put your name in a comment line at the beginning of your file.