CS 340.001: Algorithms and Data Structures  
Programming Assignment 1  
Due on Dropbox by 1:00 PM on Monday, January 31, 2011

Assignment Objectives:  
• Test understanding of class templates, linked lists, and subclasses.  
• Apply recursion to a list problem.

In this assignment, you will begin by implementing a subclass of an existing linked list class (provided on the course website) to enable the derived class to be used to recursively determine whether a list of integer values has a majority element, i.e., an element that populates over half of the list. You will also implement a driver program in order to test the derived class, using external input files of integer values as test data.

The recursive algorithm for finding the list's majority element (if it exists) follows a two-step approach:

1. Narrow the list down to a single candidate for the value that might be the majority element.
2. Sequentially count the number of occurrences of the candidate element, confirming whether or not the candidate is actually the majority element.

The first step is handled recursively. First, set up a separate empty list. Then do a pairwise comparison of the original list's elements (i.e., compare the first and second elements, the third and fourth elements, and so forth), to see if the pair of elements have the same value. If so, insert that value into the new secondary list. When finished with this step, there are three possibilities:

1. The secondary list is empty. In this case, if the original list was odd in length, the only possibility for a candidate element would be the last element in the list (which was never part of the pairwise comparisons), so use it as your candidate. Otherwise, there is no candidate, which means that there will be no majority element.
2. The secondary list contains exactly one element, which should then be used as the candidate.
3. The secondary list contains more than one element, in which case recursion should be invoked and the candidate will be whatever candidate is calculated for the secondary list.

As an example, consider the following list of 23 integer values:

2, 3, 3, 7, 3, 5, 3, 7, 3, 5, 3, 7, 3, 5, 3, 3, 7, 3, 5, 3

The first pass through this list yields the following secondary list:

2, 3

This list, however, yields an empty secondary list. Since it is even in size, it returns no candidate to the original list. Since that list was odd, however, its last element, 3, becomes the only candidate, and, in fact, a sequential count confirms that 3 is actually the majority element.

Zip-compress your entire program folder and place it on your dropbox by the deadline mentioned above. Several files of test data are available on the course website.

You must write your own code on this assignment, with adequate explanatory documentation (i.e., a paragraph at the top of each program file, including your name and a description of the file's contents, and a sentence preceding each function, describing the function's purpose).

Obtaining code assistance from any outside source is considered academic misconduct.

The only person permitted to see your code prior to the assignment deadline is the instructor.