Homework 2: Due: Monday Feb. 20 by 11:59PM via email to grader 
(dkc237@nyu.edu)

In this assignment we will slightly modify the linked list class we have been writing in class, and use it to implement a new sorting method. The new method is called \textit{radix sort} - it is one of a special class of sorting algorithms that can be used when you know the range of integers to be sorted.

Here is a short example of radix sort. There will be more extensive examples in recitation. Suppose we have an initial array \(a\) with numbers in the range \([0..99]\). For an example suppose \(a = [2,6,12,82,10,47,16]\). The numbers will be sorted based on least significant to most significant digit, in this case the ones and tens digits, so for this example there will be 2 passes. The numbers are sorted by putting them in buckets (really, on a list) corresponding to the digit in the column in question, as in the figure below:

\[
\begin{array}{c|cccccccc}
\text{buckets} & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
\hline
10 & 2 & & & & & & & 6 & 47 \\
12 & & & & & & & & 16 & \\
& & & & 82
\end{array}
\]

The numbers are then put back from the lists into the array in order, giving \(a = [10,2,12,82,6,16,47]\).

On the second pass the numbers are put into the linked lists based on the tens digit. This looks like:

\[
\begin{array}{c|cccccccc}
\text{buckets} & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
\hline
& 2 & 10 & & 47 & & & & 82 & \\
& 6 & 12 & & & & 16 & & &
\end{array}
\]

When the numbers are re-assembled into the array, they come out sorted (make sure you understand why).

Note that for this algorithm we only need to insert numbers at the end of the lists. You should therefore write a method \texttt{addRear} for the \texttt{MyList} class, along with the \texttt{addFront} already written in class. To be able to do this efficiently, you should add an instance variable \texttt{rear} to \texttt{MyList}, which refers to the last element in the list, and update
it appropriately in \texttt{addRear}. You will also need a routine to \texttt{clear()} the lists, so you can start with empty lists for each pass of the algorithm.

The next step is to implement the radix sort algorithm. You should use an array of \texttt{MyLists<Integer>} for the buckets. The \texttt{i}th entry in the Buckets array will hold the list for digit \texttt{i}. To help you organize this program, I suggest you write the simpler methods \texttt{clearBuckets}, \texttt{putArrayOnLists} and \texttt{putListsInArray}. A \texttt{printArray} method might be an early helpful method to write as well.

Let the array \texttt{a} be of size \texttt{a.length}, where this is a very small number until your code is debugged. Let the \texttt{RANGE} of the numbers also be a \texttt{final int}, set initially to 10 then 100 (with the number of passes correspondingly 1 then 2 - how do you set the number of passes?) You also need to figure out (or remember for those who did it last semester) how to extract the digit in the \texttt{d}th column from the integer using mod and integer divide operations. Initially you might debug your code using an array \texttt{a} you initialize with fixed numbers. Then test it by initializing the array using the \texttt{nextInt} method from the random number generator from the \texttt{java.util.Random} class.

Happy programming. This assignment will really help you understand the basics of generics. (which can be very tricky). I recommend compiling and debugging your code every few lines. Don’t leave it to the last day.

\textbf{Honors variation:} The honors students should do almost the same assignment with these following changes. First, you will sort an array of words. You can assume the words all have lower case letters, but they are not all the same length. Note that the word “do” comes before “dog” and after “cat”. In addition, the number of passes that radix sort needs should be calculated on the fly from the data. In other words, there is no \texttt{RANGE} that determines the number of passes in advance. You can initialize your string array in main, or read in words from a file, but again test on small data before you use bigger arrays.

For the really ambitious, an extra credit option is to write a generic radix sort class that can sort both \texttt{Integers} and \texttt{Strings}, for example. This is much harder, and involves writing an interface to supply the information that the radix sort needs. See me for details (and for the interface code) if you’re interested.