Computer Science 112 - Fundamentals of Programming II
Lab Project 8
Due on Github 11:59pm Monday 11 November

Implement the BucketSort algorithm we described in class. Write **bucketsort.py** containing (for now) a single function **`bucketSort`** (no class definition necessary), which will sort (mutate) the integer list you pass to it.

Your **`LinkedList`** class will be helpful for implementing the buckets. Instead of using the code from the slides, I found it easier to code up **`bucketSort`** myself from first principles, avoiding the **Arrays** class and other complications from the lecture slides (Lecture 29, Slide 18). Instead, I used an ordinary Python list for my array and followed the general algorithm in the slide.

Once you've tested your **`bucketSort`** function on some integer lists you build by hand (make sure to include some duplicates!), it's time to try it out on a list of random integers. To help with these, write a function **`randints(n)`** that accepts a numbers *n* and returns a list of *n* random integers in the interval [0, *n*-1]. Test your **`bucketSort`** again on a small list created by using this function.

Finally, as you may have expected, it's time to run some speed contests! Since the claim is that BucketSort can be faster than QuickSort, we'll compare these two sorting algorithms. Copying your **sorts.py**, **counter.py**, and **tools.py** files from Lab #3 will give you access to QuickSort, by adding **`from sorts import quickSort`** at the top of **bucketsort.py.** To compare the two sorting functions, use your **`randints`** function to generate a large list of random numbers. Then make a copy of this list, so you can give one copy to **`quicksort`** and the other to **`bucketSort`**. In reporting the time values, I avoided using **`int()`** to convert the elapsed time to a whole number of seconds; otherwise, the differences were too small to notice.

As I discovered through some googling, Quicksort is the best choice in general, but BucketSort can win when you have a small list. To figure out what size list counts as "small", run some experiments with different values of *n*. For example, you could start with $n = 1000$ and double *n* until you seem some interesting results.

To finish up, summarize your results in a plot using Excel (or Numbers on a Mac, or whatever spreadsheet program you like). Your plot should show time in seconds against list size *n*, with a cross-over for some value of *n.* Include your plot as a PDF in your submission. Another extra-credit opportunity would be to have your **bucketsort.py** program create a .CSV file that you can open in a spreadsheet program, so you don't have to copy the numbers into the spreadsheet by hand.