CISC106 Spring 2013 Lab08

- This lab an all subsequent labs will be due Thursday at 11:55 PM EDT on Sakai.
- The preparation problems below are to develop your understanding without creating extra work for you or the TA; these problems will not be graded. Be sure to read and understand them - they will help with the problems you must submit for grading
- Review the code examples from your notes in class.
- You may work in pairs on your lab. If you do, **one** of you should be designated to submit the assignment on Sakai. **Both of your names** should appear on code that you develop together[1].
- Whom do you think deducts more points: a happy TA, or a frustrated TA? Make your work easy to read! It isn't just good software engineering, it is good for your grade!
- EVERY MATLAB program/function must include header, doc string that contains a human-readable desciption of what the function does, and must be followed by a good series of tests, as discussed in class. Always test boundaries. Do not test erroneous input (e.g. a factorial function does not need to correctly handle strings).
- EVERY .m file must have a comment line at the very top containing your name(s), lab section, and a brief description of what the file is.

Programs (to be graded)

1. Implement the simple function `squared` which simply takes a number and squares it. Then implement the simple function `nlogn` which takes a number n and returns the value $n \cdot log_2(n)$. Note that MATLAB provides a `log2` function which gives the log_2 of its argument.
2. Now implement a function which plots both of the functions from part 1 *on the same graph*. You should plot the values in the range $100 : 100 : 1000$. Take a screenshot of the graph that gets produced and submit it with your lab. **Hint:** To get both lines to plot on the same graph, you should give the `hold all` instruction between the two calls to `plot`. If you need further direction, refer to the examples from class: `http://www.udel.edu/CIS/106/keffer/13S/examples/april4/`
3. Download *sort_timer.m* and *graph_times.m* from the course web site. Open *sort_timer.m* and notice there is a function of the same name which contains some functions for doing a *merge sort*.[2] Look beneath those functions to see some code which builds a list of random numbers (the size of the list being the parameter to `sort_timer`) and then merge sorts the list. Then note how it keeps track of the number of steps the merge sort took to sort the list. Your job is to add functions for insertion sort and modify the timing function so that it times both merge sort and insertion sort. Make sure you set it up so it correctly returns a list with merge sort's steps and insertion sort's steps. **NB:** A step should be counted for *every* comparison made during the insertion sort. **Hint:** You probably should consider using your `insert_in_order` function from lab07 in insertion sort.

[1] If you would like to work with someone but don't know whom, your TA may be able to help connect you to other students looking for lab partners.

[2] Make sure you pay attention to the comments in this code!

4. Now open *graph_times.m*. Notice that it runs sort_timer with list sizes from 100 to 1000 in increments of 100 (so you'll get the amount of steps to sort lists of size $100, 200, 300, \ldots, 1000$.) Modify it so that it plots the trendlines for both merge sort and insertion sort. Take a screenshot of the graph that gets produced and submit it with your lab.

You should submit your sort_timer.m, graph_times.m, the .m files from parts 1 and 2, the screenshots of the graphs you created in parts 2 and 4, and any other docs required by your TA on Sakai.