CISC106 Fall 2011 Lab08

- This lab an all subsequent labs will be due Sunday 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 with one or two other people on your lab (max size is three!). These people must be in your same lab section. If you do, **one** of you should be designated to submit the assignment on Sakai. **All 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 python 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 .py 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.
- Write the tests first! Real software engineers do this for very good reasons - so should you!

Preparation (do not submit for grading)

1. type the following into the shell and examine the output:

```
>>> int('stuff')
```

   You should get a `ValueError` exception with some message about `'stuff'` not being a base 10 literal.

2. Now Write a test in lab08_tests.py with the following assertion in it:

```
self.assertRaises(ValueError, int, 'stuff')
```

   You should be able to run this test and watch it pass.

[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.

Programs (to be graded)

1. Download *lab08.py* from the course web site. Open it and notice there is a class `SortTimer` with functions for merge sort and timing merge sort. You should add to this class a function for insertion sort and modify the timing function so that it times both merge sort and insertion sort. Run the timing function a few times in the shell and see what the results are. **Hint:** You probably should consider using your `insert_in_order` function from lab05 in insertion sort. Don't forget to modify it so that it increments *steps* after every comparison!

2. Ackermann's function (named after mathematician Wilhelm Ackermann) is a recursively defined function with some properties of interest to theoretical computer science. The function is defined as such:

$$A(m,n) = \begin{cases} n+1 & \text{if } m = 0 \\ A(m-1,1) & \text{if } m > 0 \text{ and } n = 0 \\ A(m-1, A(m,n-1)) & \text{otherwise} \end{cases} \quad (1)$$

   Write a function called `ackermann` inside lab08.py which implements Ackermann's function. When you write your test for the function, see `http://en.wikipedia.org/wiki/Ackermann_function#Table_of_values` for a table of input/output sets. *N.B.* that Ackermman's function will raise an exception for most pairs (m, n) where $m \geq 4$. You should have at least one assertion in your test that covers a case where an exception is raised.

You should submit your lab08.py, lab08_tests.py, a dump of the output of three separate runs of `time_sorts` and any other docs required by your TA on Sakai.