CISC106 Summer 2011 Lab02

- 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

Preparation (do not submit for grading)

1. Run the Python shell by simply running `python-2.7` with no arguments. at the prompt type in:[1]

   ```
   >>> min(5, max(1, 7))
   ```

   This is a simple example of function composition: taking the result of one function and using it as input to another function.

2. Type the following into the shell window (you need to press enter twice at the end of the function to get the prompt back):

   ```
   >>> def plus1(x):  return x + 1
   ```

   Figure out how to compose multiple calls to the `plus1` function to get the number 3 if you start by calling `plus1(0)`.

Programs (to be graded)

1. Download *lab02.py* and *lab02_tests.py* from the Labs folder on the course website. The first function in lab02.py is named `messed_up`. Without removing or adding any lines of code to the function, fix all the errors (and rename the function appropriately) so that the first test in lab02_tests.py passes. You can run your tests by issuing the command `python-2.7 lab02_tests.py` at the UNIX prompt.
2. Note the functions `f`, `g` and `h`. Also note that the tests `test_composition1` and `test_composition2` are still failing (make sure to look carefully at the error messages and understand what they are saying!)

   (a) Correct `test_composition1` by replacing the second parameter 0 in the `assertEqual` to the correct output.

   (b) Complete the function `composition2` by replacing the 0 in the return statement. Use only a composition of function calls to `f`, `g` and `h` (you can call each more than once) with x so that the asserts in `test_composition2` pass.

3. In physics, an object floats if its weight is less than or equal to the weight of the liquid it displaces. Add the following functions to your lab02.py, adding tests to lab02_tests.py for each one before moving on to the next (Think about some easy cases you can calculate by hand that you can use for test cases.)

   (a) Write a function, `total_weight`, that calculates the weight of a liquid given its volume (in cubic centimeters) and its density (in grams per cubic centimeter.) You can use the following values in your tests for liquid densities:

[1] NB: from henceforth, I will preceed any input meant for the Python shell by the Python prompt, `>>>`.

```
WATER_DENSITY = 1.0
MILK_DENSITY = 1.03
GASOLINE_DENSITY = 0.7
```

if you want, you can look up and use other densities as well.

(b) Assume you have a cubic box that is $100cm \times 100cm \times 100cm$. The box itself weighs $500g$ and is open on the top. Write a function, `max_contents` that takes as a parameter a liquid density (in grams per cubic centimeter) and calculates maximum weight of contents that could be placed in the box such that it still floats in the liquid. Your `max_contents` function should use your `total_weight` function.

(c) Write a function, `box_orders`, that takes a parameter for liquid density and a parameter for a total weight of inventory. Your function must return the minimum number of boxes required that will keep all of the inventory afloat. You may assume that you can split the inventory into the boxes evenly. However, you cannot order a fractional number of boxes - you will need to round up your order to the closest number of boxes. **Hint:** You can use the built-in `ceil` function in the math module of Python to do this, just remember you need to import the math module if you do!

You should submit your lab02.py and your lab02_tests.py and any other docs required by your TA on Sakai.