CISC106 – Lab 02

This lab is due 06/16/2013 at 11:55 p.m. (submission via Sakai)

- **Please do all of the following problems in ONE file named lab2.py. This is an INDIVIDUAL assignment, please do all work accordingly.**
- **Use comments to separate your program for each problem. For questions where you should write your answers, envelop them as comments.**
- **For now on we will be using the Design recipe to write functions and the assertEqual function to run our test cases. You need to provide at least 3 test cases for each new function you write. Please keep in mind that those elements are worth at least half of the question, so you may not want to forget them!**
- **Make sure the file cisc106.py (attached) is in the same directory of your lab2.py while running it. Also make sure you include it in your lab2.py program. You can do this by adding the following line at the very beginning of your file.**

  from cisc106.py import *

- **The problems are worth 90 points + 10 points for attending the lab session.**

Problem1: (20 points)

Given the following Python function and code:

```
1.  def  divide_by_5(number):
2.          return (number / 5)
3.  divide_by_5 (3)
```

Answer the questions (copy and paste the questions and add your answers as comments on your lab2.py file):

a. (4 pts) What is the function name?

b. (4 pts) What is the input type?

c. (4 pts) What is the output type?

d. (4 pts) What is the instruction (code) for calculating the function?

e. (4 pts) What does line 3 do?

Problem 2: **(5 points)**

In the Python Shell window, type in:

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

This is a simple example of function composition (or nested function calls): taking the result of one function and using it as input to another function. Type the following function into your lab2.py file:

```
def plus1(number):
    return number + 1
```

How should you compose multiple calls to the plus1 function to get the number 3 if you start by calling plus1(0)? On the lab2.py file, write only one statement that will provide the result 3 and print it to the screen.

Problem 3: **(15 points)**

Inside lab2.py, write your first python function using the **Design Recipe** showed in class. Then call it using assertEqual test cases to make sure it works. The function should perform the following calculation:

$$f(x) = (x + 3)^2$$

Problem 4: **(15 points)**

Place the following Python function definitions in your lab2.py file:

```
def f(x):
    return x ** 2


def g(x):
    return x + x + x


def h(x):
    return x / 2


def composition1(x):
    return f(g(h(x)))


assertEqual(composition1(10), 0)
```

Run your lab2.py script by choosing Run->Run Module.

Make sure that you understand the output.

a) Correct the above test by replacing the expected output 0 with the correct output. **(5 points)**

b) Complete the function, composition2, by replacing the "?" marks. Use only a composition of function calls to f, g, and h (you can call each more than once) with x so that the given tests pass: **(10 points)**

```
def composition2(x):
        return ???????
assertEqual(composition2(4), 12)
assertEqual(composition2(10), 75)
```

Problem 5: **(35 points)**

(Make sure to do all 4 steps of the design recipe for each function you create on this problem)

In physics, an object floats if its weight is less than or equal to the weight of the liquid it displaces.

a) Write a function, **total_weight**, which 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 constants in your tests for liquid densities: **(10 points)**

```
WATER_DENSITY = 1.0
MILK_DENSITY = 1.03
GASOLINE_DENSITY = 0.7
# see http://www.engineershandbook.com/Tables/densities.htm for more information
```

b) Assume you have a cubic box that is 100cm by 100cm by 100cm. The box itself weighs 500grams 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. **(10 points)**

Attention: 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 ceiling function in the math module of Python to do this, look it up online or in your textbook. **(15 points)**