CIS 130 - Intro to Programming - Spring 2007
Homework Assignment #3 - **INDIVIDUAL** assignment

Homework #3 DUE**: BEGINNING** of class, Wednesday, February 21, 2007

Purpose: get practice with named constants, functions that call other functions, a boolean function, and some functions that include if-statements.

How to turn in: use the tool **~st10/130submit** on cs-server to turn in the files **hw3.py** and **hw3.txt** that you create below.

NOTE: **<u>from here on out</u>**, you are expected to use the **design recipe** for ALL functions (and so you are expected to turn in contracts, purpose statements, and examples for all functions as well). You will receive significant partial credit for any function with a correct contract, purpose statement, and examples, even if it doesn't work! (And, analogously, you will lose significant credit if you don't include a correct contract, purpose statement, and examples, even if it DOES work...)

(You are also expected to use named constants whenever appropriate, even if you are not reminded to do so.)

Type all of the functions for this assignment in a file **hw3.py**.

1. [The purpose of this problem is design recipe practice, named constant practice, and practice in having functions call other functions.]

Do Exercise 3.3.1 in the CIS 130 - "HtDP" Reading Packet. Notice that this question is asking you to write **10** functions.

Also note: why so many? Because this problem is not only providing function practice, it is also intended to give you practice writing functions that appropriately call other functions. That is, if your last four functions are not extensively calling the first six functions, you are off track...

2. Write a function **worked_overtime** that expects one parameter, the number of hours worked. It should return **bool** True if number of hours worked is strictly more than 40, and it should return **bool** False otherwise. Be especially careful with the examples - how many should there be (based on your reading and the in-class discussions), and what should they include?

3. Now, **use** the function **worked_overtime** in another function, **overtime_owed**. This function expects, as its parameters, the number of hours worked, and the overtime rate of pay. [In this particular scenario, then, different people can receive different overtime rates of pay]. If the number of hours worked exceeds 40, then **overtime_owed** should return how much overtime wages are due in this case --- overtime wages are computed by multiplying JUST the hours over forty by the overtime rate of pay. But, if no overtime was worked, then the function should return 0 (no overtime is owed in this case).

For example, **overtime_owed(43, 12.50)** == **37.5**, because there are 3 overtime hours at 12.50 overtime per hour. But, **overtime_owed(37, 15.00) == 0**, because no overtime wages are owed for this case.

While it is quite possible to write this without an **if**-statement, for practice's sake you are required to appropriately use an if-statement in your solution (and you are required to appropriately call **worked_overtime** as well).

4. Now write a function **reg_pay_owed**. It expects the number of hours worked and the regular rate of pay, and returns the amount of non-overtime-pay owed. (Note, thus, that it returns only the amount of pay for the

FIRST 40 hours of work -- it doesn't for the rest, since that amount will be paid for as overtime.)

Even though this can be done without using an **if**, you are required to write yours using an appropriate **if-**statement, and appropriately calling **worked_overtime**.

5. Finally, write a new version of **gross_pay**: it takes the hours worked and the wage, and it still computes the pay owed based on paying that wage for the first 40 hours and time-and-a-half for any hours over 40 --- BUT, now it should do so by appropriately calling **overtime_owed** and **reg_pay_owed**. [Be careful -- based on what **overtime_owed** expects, what should the arguments to that function be within **gross_pay**?]

Note that this one doesn't need an **if**-statement! [You are practicing writing a function that calls other functions, here.]

6. (Adapted from Keith Cooper's section of Rice University's COMP 210, Spring 2002)
Conditionals (and Pizza Economics)

In class, we have developed a function intended to deal with some of the economic aspects of eating pizza. Our earlier function, however, ignores an important consequence of increased pizza consumption –-the need for additional exercise.

Develop a function **work_out** that computes the number of hours of exercise required to counter the excess fat from eating pizza. **work_out** expects as its parameter a number that represents daily pizza consumption, in slices, and returns a number, in hours, that represents the amount of exercise time that you need.

For a daily intake of :	You need to work out for :
0 slices	1/2 hour
1 to 3 slices	1 hour
>3 slices	1 hour +1/2 hour per slice above 3

When you are happy with your files **hw3.py** and **hw3.txt**, type the following command at the cs-server prompt:

~st10/130submit

Then follow its directions to submit your files .