**Introduction to Computer Science II**

**Lab 4.5**

Note: You do not need to submit anything this week.

**Problems**

**Implement the solutions for problems 1 and 2 in**[lab.py](http://reed.cs.depaul.edu/lperkovic/csc242/homeworks/lab.py)**.**  
  
**Lab solutions:**[**labSol.py**](http://reed.cs.depaul.edu/lperkovic/csc242/homeworks/labSol.py)  
 **1.***First*, develop a class Worker supporting just a constructor:

* \_\_init\_\_(self, name): constructor used to initialize worker name.

*When done*, develop classes HourlyWorker and SalariedWorker as subclasses of Worker.   
  
Class HourlyWorker will have methods:

* \_\_init\_\_(self, name, rate): extends Worker constructor to initialize name *and* hourly pay rate. (Note: you should make use of the Worker constructor.)
* pay(self, hours): returns the weekly pay for the worker. Hourly workers get paid the hourly rate for the actual number of hours worked, for hours up to 40; if hours is more than 40, the excess hours are paid at time and a half.

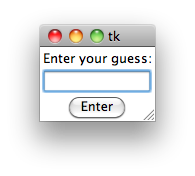
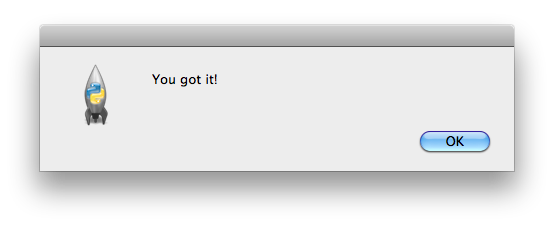
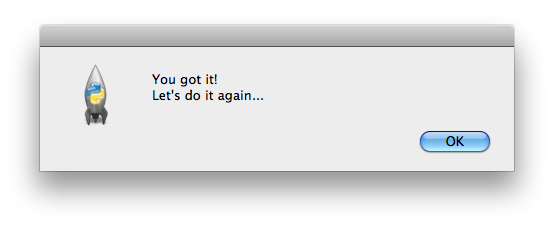
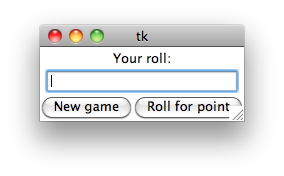
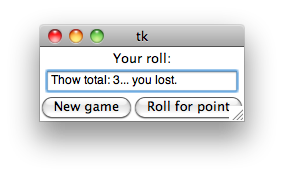
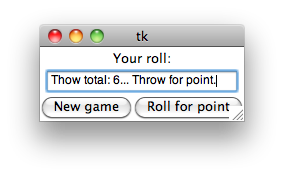
Class SalariedWorker will have methods:

* \_\_init\_\_(self, name, salary): extends constructor to initialize name *and* weekly salary. (Note: you should make use of the Worker constructor.)
* pay(self): returns the weekly salary for the worker.

Usage:  
>>> workerA = HourlyWorker('Alice', 35)  
>>> workerA.pay(65)  
2712.5  
>>> workerA.pay(30)  
1050  
>>> workerA.pay(45)  
1662.5  
>>> workerB = SalariedWorker('Bob', 1400)  
>>> workerB.pay()  
1400  
>>> workerC = Worker('Carol')  
>>> workerC.pay()  
Traceback (most recent call last):  
  File "<pyshell#10>", line 1, in <module>  
    workerC.pay()  
AttributeError: 'Worker' object has no attribute 'pay'

**2.    [10 points]** Develop a class myList that is a subclass of the built-in list class. The two list methods your myList class should override are the sort method and the len operator. So myList objects should behave just like regular lists, except as shown below:

Usage:  
>>> lst = myList([5,4,8])  
>>> lst[0]  
5  
>>> lst.reverse()  
>>> lst  
[8, 4, 5]  
>>> lst.sort()  
You wish...  
>>> lst  
[8, 4, 5]  
>>> len(lst)  
17

Note: Function len returns the sum rather than the length of the list.  
  
  
  
**The solution to each problem below should be in a separate Python module (file).**  
  
  
**3.**Develop a GUI app that implements a number guessing game. The GUI should start by choosing a random number between 0 and 9 (using the randrange function from the random module) and then opening up a GUI which allows the user to enter number guesses:  
  
  
Your GUI should have an Entry widget for the user to type the number guess and a Button "Enter" widget to enter the guess. If the guess is correct, a separate window should inform the user "You got it!":  
  
The user should be able to enter guesses until he/she makes the correct guess. Make sure to make your GUI user friendly by erasing the guess each time the user presses the "Enter" button: this means that the user will not need to erase the old guess in order to enter a new one.  
  
  
**5**.    Now modify the above app so that a new game starts when the user has guessed the number. The window informing the user that she made the correct guess should be something like:  
  
Note that a new random number would have to be chosen at the start of each game.   
  
  
  
  
**5.**Write a GUI app that implements the Craps gambling game. When started, a GUI window should appear:  
  
You start a new game of Craps by clicking the button "New Game". The result of the initial roll is then shown in the Entry widget:  
  
If the initial roll is not a win or a loss, such as in:  
  
the user will have to click the button "Roll for point", and keep clicking it until he/she wins.  
