2013 BHCSI Object-Oriented Design in Python
Test #2 (100 points) Solutions
7/24/2013

1) (15 pts) Complete the program below so that it reads in a sentence entered on a line by itself by the user and reports the number of words in the sentence and the length of the longest word in the sentence.

```
def main():

    sentence = input("Please enter your sentence.\n")

    tokens = sentence.split()      # 3 pts
    numWords = len(tokens)         # 3 pts

    numChars = len(tokens[0])           # 1 pt
    for i in range(len(tokens)):        # 3 pts
        if len(tokens[i]) > numChars:   # 3 pts
            numChars = len(tokens[i])   # 2 pts


    print("Your sentence has", numWords, "words.")
    print("The longest word has", numChars,"characters.")

main()
```

2) (10 pts) The Lucas numbers are defined as follows: $L_1 = 1$, $L_2 = 3$, $L_n = L_{n-1} + L_{n-2}$ for all $n > 2$. (Thus, the next few Lucas numbers are 4, 7, 11 and 18.) Write a recursive function that takes in n and returns the n^{th} Lucas number. You may assume n is a positive integer.

```
def lucas(n):

    if n == 1:                              # 1 pt
        return 1                            # 1 pt
    elif n == 2:                            # 1 pt
        return 3                            # 1 pt
    else:                                   # 1 pt
        return lucas(n-1) + lucas(n-2)   # 5 pts
```

3) (12 pts) Complete the following function that takes in a list of numbers and returns the product of all of the numbers in the list.

```
#Takes a list of numbers and returns the product
def multiply(numbers):

    product = 1                              # 2 pts
    for i in range(len(numbers)):            # 4 pts
        product = product*numbers[i]         # 4 pts

    return product                           # 2 pt
```

4) (12 pts) What is the output produced by the following code segment:

```
def func(n):
    if n < 5:
        print(n, end=" ")
    else :
        print(n%5, end=" ")
        func(n//5)

func(67)
print()
func(116)
```

2 3 2
1 3 4 (2 pts each number)

5) (10 pts) What will the following code segment print (Don't worry about ordering for the last line)?

```
x = {'Pie':'Dog', 'Bro': 'Bacon', 'John':'Asia', 'Eggs':'Ralph'}

if 'Ralph' in x:
    print("Hello Ralph!")

if 'John' in x:
    print("John is here!")

print(x['Pie'])

for i in x:
    print(i, end = ' ')
```

John is here! # 2 pts, 2 pts for not having Ralph
Dog # 2 pts
Bro Eggs John Pie # 1 pt each

6) (13 pts) Complete the recursive function below that it prints all odometer readings of length n that have digits in strictly increasing order with the prefix current.

```
def printIncOdometer(n, current):

    if len(current) == n:
        print(current)
        return

    start = 0
    if len(current) > 0 : # 1 pt

        start = int(current[len(current)-1])+1 #6 pts

    for i in range( start ,10):  # 2 pts

        printOdometer( n , current+str(i) )
                    #1pt  3 pts
```

7) (25 pts) Write a function that takes in two dictionaries that map items to prices and returns a new "merged" dictionary that contains each item in either dictionary. If an item is contained in exactly one of the dictionaries, keep its price as stated. If an item is contained in both dictionaries, in the merged dictionary set its price to the minimum of the two prices. Remember, you should NOT make any changes to either of the two existing dictionaries. Instead, you'll create a third new dictionary and add to it all unique items from the two input dictionaries, setting the prices as previously mentioned.

```
def makeMergedPriceList(priceListA, priceListB):

    ans = {}                                # 2 pts

    for item in priceListA.keys():          # 4 pts
        ans[item] = priceListA[item]        # 4 pts

    for item in priceListB.keys():          # 4 pts
        if (not item in ans.keys()) or priceListB[item] < ans[item]:#6p
            ans[item] = priceListB[item]    # 4 pts

    return ans                              # 1 pt
```

8) (3 pts) What items are primarily sold at Books-A-Million?

Books (3 pts)