Test 2 Topics for MCS-177 (Fall, 2013)

Reminders:

1. If you need an alternative testing time or location, set that up with me as soon as possible.

2. You may use a single 8 1/2 by 11 sheet of paper with hand-written notes for reference. (Both sides of the sheet are ok.)

3. The help sessions for the week before test 2 are scheduled as follows:

   - Professor Yu's regular office hours are MWF from 1.30pm – 2.30pm.
   - The tutors (Kevin Dexter and Nate Jenson) offers help sessions for test 2 on Wednesday (11/6) from 7pm – 9pm, Thursday (11/7) from 7pm – 9pm, and on Sunday (11/10) from 6pm – 9pm.
   - Professor Yu will hold two extra help sessions on Saturday (11/09) from 6pm – 9pm and on Sunday (11/10) from 6pm – 9pm.

Topics:

- List operations like string operations: len, +, *, [], [:], in, and for
- Table 4.1 on page 122, Table 4.2 on page 125
- List modification: []=, append, sort, index, count, remove, split, join
- List comprehensions: [<*expression*> for <*item*> in <*sequence*>] and [<*expression*> for <*item*> in <*sequence*> if <*condition*>]
- Dictionaries: {}, {*key*:*value*, ...}, [], []=, in, list, len and for
- Table 4.3 on page 137
- Opening and closing files in Python (table 5.1 on page 156)
- Using for loops to iterate through each line of the file (Listing 5.1 on page 158)
- Writing a file
- Table 5.4 on page 164

- String formatting
- Table 5.2 and Table 5.3 on page 162
- Looping subject to a condition: while
- Image processing: nested loops, pixels, and images
- Table 6.1 on page 186, Table 6.2 on page 187, Table 6.3 on page 188, Table 6.4 on page 189, Figure 6.2 on page 191

Sample Questions:

1) In the following Python session, there are ten spots where an expression is evaluated, but instead of the value being shown, an italic *a* through *j* is shown. Indicate what each of these ten values would be.

```
>>> n=[10,20,30]
>>> m=[40,50,60]
>>> s=n+m
>>> t=n
>>> s[0]
 a
```

```
>>> n[0]=100
>>> max(s)
b
```

```
>>> t.append(1000)
>>> len(n)
c
```

```
>>> m[-1]
d
```

```
>>> t.sort()
>>> n
```

e

```
>>> m[1:2]
```
f

```
>>> t[0]
```
g

```
>>> 10 in n
```
h

```
>>> 10 in s
```
i

```
>>> [x+5 for x in m]
```
j

2) Consider the following two functions:

```
def divTwo(k):
    count=0
    while k>0:
        k=k-2
        count=count+1
    return count

def tens(n):
    count=0
    while n%10 == 0:
        n=n//10
        count=count+1
    return count
```

(a) What is the value of `divTwo(2)`?

(b) What is the value of `divTwo(4)`?

(c) What is the value of `divTwo(3)`?

(d) Are there any values of `k` for which `divTwo` does no subtractions? If so, given an example.

(e) Are there any values of `k` for which `divTwo` will go into an infinite loop? If so, given an example.

(f) What is the value of `tens(170)`?

3) What are the four values that are printed by the following program?

```
nums = [1,2,3]
others = [4,5,6]

def f(nums):
    nums.append(10)
    return nums

print(f(others))
print(nums)
print(others)

x = 5

def g(x):
    return h(x*10)

def h(y):
    return x+y

print(g(x+1))
```

4) Assume that there is a procedure called `factorial` that follows the contract below.

```
# factorial: integer -> integer
```

The procedure factorial takes a number n and returns the value of n!. For example, `factorial(3)` returns 6 while `factorial(0)` returns 1. You are working on a bigger program and you keep finding out that you use `factorial` many times. It looks inefficient to you to have to recalculate the factorials after you have used it before. So you decide to build a dictionary that contains the values of factorials.

(a) Define a global variable called `factorialValues` that is an empty dictionary.

(b) Write the contract and implementation of `lookUpFactorial` that takes a number and returns the factorial of the given number. `lookUpFactorial` should check if the factorial of the given number can be looked up in the dictionary first. If it is not found in the dictionary, it should calculate the corresponding factorial value and insert it into the dictionary so that next time you look up the factorial of the same number, it does not have to calculate it again.

```
>>> lookUpFactorial(1000)
```

`40238.....` (this takes a while to compute)

```
>>> lookUpFactorial(1000)
```

`40238.....` (comes out immediately)

5) Write the contract and implementation of a function `formatRainData` that read in the `rainfall.txt` file from textbook page 156, and then write to a new file called `rainfallfmt.txt`. The new file should format each line so that the city is right-justified in a field that is 25 characters wide, and the rainfall data should be left-justified in a field that is 5 characters wide.

6) Use a while loop to implement the for loop `for i in range(10)`. Next, use a while loop to implement the for loop `for i in range(10,-1,-1)`.

7) Jimmy wants to write an image filter for Instagram to impress his hipster friends. Jimmy's idea is to take a photo and increase the red intensities of each pixel (he calls it "the red filter").

(a) First, write the contract and implementation of a function `doubleRed` that take in a pixel and double the red intensity of that pixel (recall that every color is composed of various intensities of red, green, and blue – the RGB values).

(b) Next, write the contract and implementation of a function `redFilter` that takes in an image object and double the red intensity of each pixel in that image.