COSC 101, Exam #2
14 March 2014

Name: _____ Section: 9:20 / 10:20

Write your name and circle your section. Do not open the exam until instructed to do so.

You have 50 minutes to complete this exam.

There are 5 questions and a total of 46 points available for this exam. Don't spend too much time on any one question.

When defining functions, it is not necessary to write docstrings nor is it necessary to write comments.

Since indentation is important in Python, please be sure that your use of indentation is obvious for any code you write.

If you want partial credit, show as much of your work and thought process as possible.

If you run out of space for answering a question, you can continue your answer on one of the scrap pages at the end of the exam. If you do so, be sure to indicate this in two places: (1) below the question, indicate which scrap page contains your answer, and (2) on the scrap page, indicate which question you are answering.

Question	Points	Score
1	7	
2	6	
3	10	
4	13	
5	10	
Total:	46	

1. (7 points) Assume that the following statements have already been executed:

```
a = 'spring'
b = 'leaves^0'
c = [ 3, 7, 13 ]
d = [ 'base', [ a ], 'ball' ]    # note: the a is not in quotes!
```

For each of the following expressions and statements, evaluate the code and write the resulting value, or identify the error in the code that would prevent it from running.

(a) `(len(a) < (len(b) % 2)) or int(b[-1]) == 0`

Solution: True

(b) `c + d[1]`

Solution: [3, 7, 13, 'spring']

(c) `len(d) + c[-2]`

Solution: 10

(d)
```
a[-1] = c[1]
print a
```

Solution: Error: 'str' object does not support item assignment

(e)
```
c[0] = c[2]
print c
```

Solution: [13, 7, 13]

(f) `d + c[1]`

Solution: Error: can only concatenate list (not "int") to list

(g)
```
def strstuff(s):
    return s*2
    return s[:-1]

print strstuff(d[0])
```

Solution: basebase

2. (a) (3 points) What is the output of the following program?

```
a = 'corn'
def function(b):
    a = b * 2
    b = b + 'corn'
    print "A =", a
    return b

b = function('pop')
print "a =", a
print "b =", b
```

Solution: A = poppop a = corn b = popcorn

(b) (3 points) What is the output of the following program?

```
a = 'pea'
b = 'nuts'
def function(b):
    a = b + 'cob'
    b = b + 'nuts'
    print "A =", a
    print "B =", b
    return b

function('corn')
print "b =", b
```

Solution: A = corncob B = cornnuts b = nuts

3. (a) (4 points) Consider the following program that includes a nested for loop:

```
def swirly(loops):
    for this in range(loops+1):
        for that in range(4-this):
            print that+1,
        print

swirly(2)
```

What is the output of this program?

Solution:

```
1 2 3 4
1 2 3
1 2
```

(b) (6 points) Translate the above `swirly` function to one in which all `for` loops are converted to `while` loops. The iterator variables (`this` and `that`) should take on the same values in your translated program as in the original version.

Solution:

```
def while_swirly(loops):
    this = 0
    while this < loops+1:
        that = 0
        while that < 4-this:
            print that+1,
            that += 1
        print
        this += 1
```

4. This question has two parts. The second part appears on the next page.

   (a) (10 points) Write a function `last_name` that has one parameter — a string representing a person's full name — and extracts the person's last name and returns it as a string. Given a string representing a person's full name, the last name is defined as the sequence of non-space characters following the *last space* in the full name. For example:

   - `last_name('Tom Brady')` should return `'Brady'`
   - `last_name('P. Manning')` should return `'Manning'`
   - `last_name('Russell C. Wilson')` should return `'Wilson'`
   - `last_name('Ronald Vincent "Jaws" Jaworski')` should return `'Jaworski'`

   You **must** use a `while` loop and **cannot** use a `for` loop. In addition, you **cannot** use methods on strings (such as `find`). You may use string slicing.

Solution:

```
def last_name(full):
    last = ''
    i = len(full) - 1
    while i > 0 and full[i] != ' ':
        last = full[i] + last
        i -= 1
    return last
```

This problem continues on the next page...

(b) (3 points) Write a main function that asks the user for their full name and then print some commentary about their last name. Specifically, if the user's last name is 3 or fewer characters long, it should print, `"Your last name, XX, is short."` where `XX` is replaced with the user's last name. If the name has more than 3 characters, it should print `"Your last name, XX, is not that short."` where again `XX` is replaced with the user's last name.

Solution:

```
def main():
    full_name = raw_input("Enter name: ")
    last = last_name(full_name)
    if len(last) <= 3:
        print "Your last name," + last + ", is short."
    else:
        print "Your last name," + last + ", is not that short."
```

5. (10 points) This was a cold winter. One way to measure the cold is to compute *heating degree days*. Let's say we want to find out how much we need to heat a home, on average, for a given day, where our target temperature is 65 degrees Fahrenheit. If the average temperature on Monday is 15 degrees, then the heating degree for Monday is $65 - 15 = 40$. If the average temperature on Tuesday rose to 55 degrees, then the heating degree for Tuesday is $65-55 = 10$. And if by some miracle it was 67 degrees on Wednesday, the heating degree for Wednesday is 0 because we shouldn't need to turn on the heat at all.

   Write a function, `heat_needed`, that takes in a list of average daily temperatures and returns a new list that contains differences between the average daily temp and 65, *but only for days that were below 65 degrees*. Examples:

   - `heat_needed([15,25,22,20])` should return `[50,40,43,45]` because we need $65-15 = 50$ degrees of heat on day 1, and $65 - 25 = 40$ degrees on day 2, etc.
   - `heat_needed([66,65,70])` should return `[]` because we don't need to turn on the heat on any of these three days.
   - `heat_needed([30,45,66,64,40,70])` should return `[35, 20, 1, 25]` because we difference each daily temperature with 65 and *omit any difference that is zero or negative*.

Solution:

```
def heat_needed(temps):
    heat = []
    for temp in temps:
        if temp < 65:
            heat += [65 - temp]
    return heat
```

(This page is intentionally blank. Label any work with the corresponding problem number.)

(This page is intentionally blank. Label any work with the corresponding problem number.)

(This page is intentionally blank. Label any work with the corresponding problem number.)