COM110 Lab 11
Recursion and lists of lists

1. Write a recursive version of the following iterative program.

```
def getInput()
   n = eval(input("Enter a number greater than 0: "))
   while n <= 0:
        n = eval(input("Enter a number greater than 0: "))
   print("Thank you.")
```
☺ **Get check 1** ☺

2. Write an iterative (using a while or for loop) version of this recursive "countdown" program from class.

```
def countdown(n):
    print(n)
    if n == 0:
        print("End of program.")
    else:
        countdown(n-1)
```
☺ **Get check 2** ☺

3. **List of lists.** Consider the following matrix of integers:

```
1     2     3     4
5     6     7     8
9     10    11    12
13    14    15    16
```

Such a matrix, or "table," would be represented in python using a list of lists (aka a 2-dimensional list).

a) To represent the above matrix, simply hard-code the following list of lists.

```
twoDnumList = [ [1,2,3,4], [5,6,7,8], [9,10,11,12], [13,14,15,16]
              ]
```

b) How would you print out the fourth item of the third sub-list?

☺ **Get check 3** ☺

c) Print out the entire 2D list of values, one value at a time, ie. 1 to 16 one number per line. One way would be to tediously type

```
print(twoDnumList[0][0])
print(twoDnumList[0][1])
print(twoDnumList[0][2])
print(twoDnumList[0][3])
print(twoDnumList[1][0])
print(twoDnumList[1][1])
...
```

Do it a better way.

☺ **Get check 4** ☺

d) Modify your code in part c so that it prints it out the numbers in the shape of a 2D matrix, as displayed at **3.** above. (I.e., one sub-list per row.) Hint: recall from page 36 of Zelle, that the print() method has an optional parameter called end, which is set to the newline character "\n" by default. This is why whenever you do a regular print() call, the output "cursor" always automatically goes to the next line afterward. The end parameter can be set to any character you'd like, e.g., end = "\t" would tack a "tab" character onto the end of whatever you are print()ing.

☺ **Get check 5** ☺

e) Write code to *generate* the following list of lists. (You will not get credit for hard-coding it in. You must use a nested loop.)

```
twoDListByRows =    [[ 0, 0, 0, 0],
                     [ 1, 1, 1, 1],
                     [ 2, 2, 2, 2],
                     [ 3, 3, 3, 3]]
```

f) Now write code to generate this list of lists.

```
twoDListByCols =    [[ 0, 1, 2, 3],
                     [ 0, 1, 2, 3],
                     [ 0, 1, 2, 3],
                     [ 0, 1, 2, 3]]
```

☺ **Get check 6** ☺

g) And now this one.

```
            multiples = [[0,0,0,0],
                         [0,1,2,3],
```

```
                         [0,2,4,6],
                         [0,3,6,9]]
```

☺ **Get check 7** ☺

h) And finally, write code to generate the original TwoDnumList.

```
twoDnumList = [[ 1,  2,  3,  4],
               [ 5,  6,  7,  8],
               [ 9,10,11,12],
               [13,14,15,16]]
```

Hint: start by generating this matrix, then simply add one to each entry

```
twoDnumList = [[ 0,  1,  2,  3],
               [ 4,  5,  6,  7],
               [ 8,  9,10,11],
               [12,13,14,15]]
```

Hint for the hint: start by generating this matrix

```
twoDnumList = [[ 0,  0,  0,  0],
               [ 4,  4,  4,  4],
               [ 8,  8,  8,  8],
               [12,12,12,12]]
```

☺ **Get check 8** ☺

i) *What's the benefit of generating these 2D arrays using nested loops rather than hard-coding them??* Modify your code so that the size of the matrix being generated is `nxn` (rather than 4x4) where `n` is an integer value input by the user!

☺ **Get check 9** ☺

Bonuses: Try any of the following options in any order.

A. (Simulation and Randomness.) Read first four paragraphs here about the Monty Hall problem: http://en.wikipedia.org/wiki/Monty_Hall_problem. Create a simulation that demonstrates switching your door choice after the host opens a door without the car *does* improve your odds of winning the car.
B. You may now take a shot at any bonuses from previous labs that you didn't complete.
C. You may also work on current assignment (assignment 5)