CMPSC-122: Intermediate Programming
Spring 2018

Lab #8

Due Date: 03/16/2018, 11:59PM

Instructions:

- The work in this lab must be completed alone.
- If you need guidance, attend to your recitation class.
- Read the "Submitting assignments to Vocareum" file for instructions on how to submit this lab
- Do not change the function names or given code on your script
- The file name must be LAB8.py (incorrect name files will get a 0 score)
- You are responsible for testing your code. Use `python -i LAB8.py` in your terminal (or command prompt) to provide input to your functions. Test with as many data as you feel comfortable
- Each function must return the output (Do not use print in your final submission)
- Do not include test code outside any function in the upload. Remove all your testing code before uploading your file. If you are using input() to insert values in your functions and print to see the values, remove them.

Exercise 1 [2 pts]. Without using the Python interpreter, what is the output of this program after the following statements are executed? Insert your answer in the function *answers()* provided in the starter code.

```
def myFunction(n):
    if n== 0:
        return 0
    else:
        return n + myFunction(n-1)

>>> myFunction(7)
```

Exercise 2 [8 pts].
NOTE: There is no partial credit for this exercise, you will get credit only if your output is correct!

The code below shows an iterative function that returns a start pattern in shape of a right triangle:

```
def triangle(n):
    out=""
    space=0

    for i in reversed(range(0, n)):
        for j in range(0, space):
            out+=" "
        space+=1
```

```
        for k in range(0, i+1):
            out+="*"
        out+="\n"
    return out

>>> triangle(5)
'*****\n ****\n  ***\n   **\n    *\n'
>>> print(triangle(5))
*****
 ****
  ***
   **
    *
```

Notice that the function **returns** the pattern as '*****\n ****\n ***\n **\n *\n', and by using the print method outside the function, the user can see the right triangle shape.

In the starter code, there is a function called triangle that calls the function *recursive_triangle*(*x*,*n*) once.

Write the recursive function *recursive_triangle*(*x*, *n*) that **returns** a string with the LAST *x* lines of a right triangle of base and height *n*.

- *recursive_triangle*(*x*, *n*) must make a recursive call, otherwise, no credit is given
- *recursive_triangle*(*x*, *n*) must return the pattern as ' ***\n **\n *\n'. You can use the print method during testing to check if your pattern is correct.
- Calling *triangle*(*n*) should return the complete right triangle.

EXAMPLES:

```
>>> triangle(6)
'******\n *****\n  ****\n   ***\n    **\n     *\n' #This is the output that will be graded, make sure your function returns the value
>>> print(triangle(6)) #You can call the print method on your function to see the triangle pattern
******
 *****
  ****
   ***
    **
     *
>>> recursive_triangle(3,6)
'   ***\n    **\n     *\n'  #This is the output that will be graded

>>> print(recursive_triangle(3,6))
   ***
    **
     *
```

```
>>> triangle(4)
'****\n ***\n  **\n   *\n'
>>> print(triangle(4))
****
 ***
  **
   *

>>> recursive_triangle(2,4)
'  **\n   *\n'
>>> print(recursive_triangle(2,4))
  **
   *
```