CSC 401 ASSIGNMENT FIVE

Due Date: Tuesday, Oct. 17th by 11:58 PM

The purpose of this assignment is to assess your understanding of

- Nested Loops
- While Loops
- Two-Dimensional Lists
- Loop Patterns
- Iteration Control Statements

SUBMISSION

- Include your full name as a comment on the first line of your Python program.
- Include the problem number as a comment before each user defined function.
- Code all the problems in one Python file (.py) labeled as YourName_HW5.py
- Upload one file to Submissions folderlabeled as YourName_HW5.py

PROBLEMS

Note: You may not use Python statements, functions, data types, etc. that were not discussed in the reading assignment or the lecture notes/video for this week or previous weeks. This is a class for students who have not programmed before and I expect every one to code on the same level. If you have a better way of writing the code, then upload two versions: one that codes according to the specifications and the other that demonstrates advanced programming techniques.

I encourage you to use computational thinking to solve the problems. These are straight-forward solutions but developing a good habit of analyzing the problem and describing the steps will serve you well as the problems get more complex.

In this assignment the solution to each problem should be coded in a user-defined function. Be sure to include a return statement in each function even is nothing is returned. All of the function should be in one file.

1. (Squares, 15 pts) Write 3 functions squareQuads(n), squareDiagonals(n) and squareTriangle(n). Each function is based on the 'hollow' square we discussed. The modifications are small but require analysis of the request to complete successfully. As we did in the lectures, use the print ('*', end='') statement to display the variations. All are based on at least a 5 x 5 square, and the dimensions must be an odd number. In the examples below I used a 9 x 9 square (n = 9)

    a. squareQuads(n) should display a n x n square with 4 quadrants.

    b. squareDiagonals(n) should display a n x n square with diagonals from opposite corners.

c. squareTriange(n) should display a n x n square with an inner triangle. The size of the triangle should vary with the size of the square. It is important that the sides of the triangle, never touch the sides of the square, i.e. there should be a one * width around the right angle sides of the triangle and a one * width from all top/right side of the square for the top of the hypotenuse and a one * width from the bottom/left side of the square for end of the hypotenuse.

Sample:

```
Quadrants
* * * * * * * * *
*       *       *
*       *       *
*       *       *
* * * * * * * * *
*       *       *
*       *       *
*       *       *
* * * * * * * * *

>>> squareDiagonals(9)
Diagonals
* * * * * * * * *
* *           * *
*   *       *   *
*     *   *     *
*       *       *
*     *   *     *
*   *       *   *
* *           * *
* * * * * * * * *

>>> squareTriangle(9)
Inner triangle
* * * * * * * * *
*               *
*   * * * * *   *
*   * * * *     *
*   * * *       *
*   * *         *
*   *           *
*               *
* * * * * * * * *
```

2. (Nested Loops, 15pts). In the lectures, we calculated the sum of the values in each sublist of a multi-dimensional list. Now you are to write a function columnSum(lst) that takes a multi-dimensional list of any size as a parameter, and returns the sum of each column in an one-dimensional list. The number of sublists and the length of each sublist must be the same, i.e. 4 rows x 4 columns. Hint (use indexes and range)

Sample:

```
>>> columnSum([[5,9,2], [3,5,7], [8,1,6]])
[16, 15, 15]
>>> columnSum([[1,15,15,4],[12,6,7,9],[23,89,34,55],[96,7,82,51]])
[132, 117, 138, 119]
```

3. (While loop, 10 pts) Write a function numLoops() that requests a positive integer from the user and executes the following algorithm. If the number is even, divide it by 2; otherwise, multiply the number by 3 and add 1. Repeat this process with the resulting number and continue repeating the process until the number 1 is reached. After the number 1 is reached, the function should return the number of iterations required to reach 1. Note: a number is even if int(num/2) == num/2. (Test program with the numbers 9,21, and 27)

4. (While loop, 10 pts). Write a function numLetters() that keeps prompting the user for words until they hit return. The function should then return the percent of 3-letter words that were entered. So, if a user entered ten words and 4 of the words had 3 letters, then the percent of 3-letter words is 40

IF YOU HAVE ANY QUESTIONS REGARDING THIS ASSIGNMENT, PLEASE POST THEM TO THE ASSIGNMENT FIVE DISCUSSION FORUM.