Programming Assignment 5

Your solution is to be written using Python 3. Make sure you provide comments including the file name, your name, and the date at the top of the file you submit. Also make sure to include appropriate docstrings for all functions.

The names of your functions must exactly match the names given in this assignment. The order of the parameters in your parameter list must exactly match the order given in this assignment.

For any given problem below, you may want to write additional functions other than those specified for your solution. That's fine with us.

Keep in mind that the point of this assignment is to give you practice with nested loops and nested lists, not to find ways to avoid them.

Problem 1

Occasionally, instructors will drop the lowest score from a series of scores on quizzes or homework assignments. Unfortunately for you, I'm not one of those instructors. On the other hand, I know instructors who would like to employ this policy but can't do the math. To encourage this sort of generosity among instructors, write a function called `printAverages` that expects a two-dimensional table of integers (implemented as a list of lists) as its only parameter, where each row of the table contains the quiz scores for a student. For each row of the table, your function should compute the average of all but the lowest score in the row and ***print*** that average. For example, given an table of quiz scores that looks like this:

row \ col	0	1	2	3	4
0	100	100	0	100	100
1	80	75	70	65	5
2	0	10	10	0	10

your `printAverages` function should print this:

```
Average for row 0 is 100.0
Average for row 1 is 72.5
Average for row 2 is 7.5
```

You may assume that no quiz score is less than 0 or greater than 100. The values may be either integers or floating point numbers. You may also assume that every row contains the same number of scores. Note that only one lowest score in a row is dropped. Your function should not ask for keyboard input, and the only value it returns is None (that's not the string "None").

Note that the table of scores could have any number of rows or columns so long as the number is greater than zero; the dimensions of the table will not necessarily be the same as in the example given on the previous page. The list of lists for the sample table would look like this:

```
[[100, 100, 0, 100, 100],
 [80, 75, 70, 65, 5],
 [0, 10, 10, 0, 10]]
```

Problem 2

Create a function named `addTables` that expects two 2-dimensional tables of integers (both implemented as a list of lists) as parameters. Both tables will have the same dimensions, so your `addTables` function does not have to verify that the tables are the same size. Your function should create a new table with the same dimensions as the two tables passed as parameters. For the remainder of this explanation of how your function should work, we'll call the two tables passed as parameters table1 and table2, and the new table will be table3.

Once table3 has been created, your function should add the value of each element in table1 to the value of the corresponding element of table2 and store the sum at the same location in table3. For example, the value stored at indexes 1,2 in table3 would be the sum of the values stored at indexes 1,2 in table1 and table2. In other words, if you think of the first two tables as matrices, this function performs matrix addition. When the individual values from table1 and table2 have been added and stored in table3, your function should ***return*** the new table, table3. Your function does not ask for keyboard input, it does not print anything, and it must not modify any of the values in table1 and table2. Here are some examples of this function's behavior:

```
>>> addTables([[2,3,5],[7,9,11]],[[13,17,19],[23,29,31]])
[[15, 20, 24], [30, 38, 42]]
>>>
addTables([[1,8],[2,7],[3,6],[4,5]],[[9,16],[10,15],[11,14],[12,13]])
[[10, 24], [12, 22], [14, 20], [16, 18]]
>>>
```

Here are a couple of graphical examples to help explain what we expect your function to do (but note that we don't want your function to display these graphs...these pictures are just here to help you visualize what the function is computing):

$$\begin{array}{ccccc}
\begin{array}{c}\text{table passed}\\ \text{to function}\end{array} & & \begin{array}{c}\text{table passed}\\ \text{to function}\end{array} & & \begin{array}{c}\text{table returned}\\ \text{from function}\end{array} \\
\begin{array}{|c|c|c|}\hline 2 & 3 & 5 \\ \hline 7 & 9 & 11 \\ \hline\end{array} & + & \begin{array}{|c|c|c|}\hline 13 & 17 & 19 \\ \hline 23 & 29 & 31 \\ \hline\end{array} & = & \begin{array}{|c|c|c|}\hline 15 & 20 & 24 \\ \hline 30 & 38 & 42 \\ \hline\end{array}
\end{array}$$

$$\begin{array}{|c|c|}\hline 1 & 8 \\ \hline 2 & 7 \\ \hline 3 & 6 \\ \hline 4 & 5 \\ \hline\end{array} + \begin{array}{|c|c|}\hline 9 & 16 \\ \hline 10 & 15 \\ \hline 11 & 14 \\ \hline 12 & 13 \\ \hline\end{array} = \begin{array}{|c|c|}\hline 10 & 24 \\ \hline 12 & 22 \\ \hline 14 & 20 \\ \hline 16 & 18 \\ \hline\end{array}$$

Where to do the assignment

You can do this assignment on your own computer, or in the labs. In either case, use the IDLE development environment -- that's what we'll use when we grade your program. Put all the functions you created in a file called "prog5.py".

Submitting the Assignment

We will be using Canvas to turn in assignments. Submit the file containing your functions as an attachment. Do NOT cut-and-paste your functions into a text window. Do NOT hand in a screenshot of your functions' output. We want one file from you: "prog5.py".

Saving your work

If you are working in the lab, you will need to copy your program to your own flash-drive. To save it on flash-drive, plug the flash-drive into the computer (your TA or the staff in the labs can help you figure out how), open the flash-drive, and copy your work to it by moving the folder with your files from the Desktop onto the flash-drive.