CS-UY 1114: Lab 5

Nested Loops

You must get checked out by your lab CA **prior to leaving early.** If you leave without being checked out, you will receive 0 credits for the lab.

Restrictions

The Python structures that you use in this lab should be restricted to those you have learned in lecture so far. Please check with your teaching assistants in case you are unsure whether something is or is not allowed!

Create a new python file for each of the following problems.

Your files should be named *lab[num]_q[num].py* **similar to homework naming conventions.**

Problem 1: *Nested Loops Galore*

Solve this problem by hand. Format the output exactly how it would be when the code snippet is run.

What would be printed for each of the following code snippets?

(a)

```
for i in range(4):
    for j in range(4):
        print(i*j, end = " ")
    print()
```

(b)

```
for i in range(5):
    for j in range(i+1, 5):
        print(i, j)
```

(c)

```
for i in range(1, 10):
    if i%2 == 0:
        for j in range(i, 10, 2):
            print(j, end = " ")
        print()
```

(d)

```
for i in range(1, 10):
    if i%2 == 0:
        for j in range(0, i, 2):
            print(j, end = " ")
        print()
```

Problem 2: *Multiplication Through Addition*

This problem is intended to get you familiar with nested loops. What better way than to do mulitplication the way it was likely taught to you all those years ago: through addition!

Your program must:

- Prompt the user for 2 factors that are integers
- Validate the user input is positive and non-zero (if they're not, output an ERROR message)
- Calculate the product of the 2 factors without using the `*` operand
- Continue the program until the user inputs `"Q"` for quit

Here is a possible output:

```
Press ENTER to calculate a product or Q to quit:
Please input your first factor: -1
Please input your second factor: 2
ERROR: Positive integers must be entered.
Press ENTER to calculate a product or Q to quit:
Please input your first factor: 101
Please input your second factor: 2
Your product is: 202
Press ENTER to calculate a product or Q to quit: Q
```

Problem 3: *Power Table*

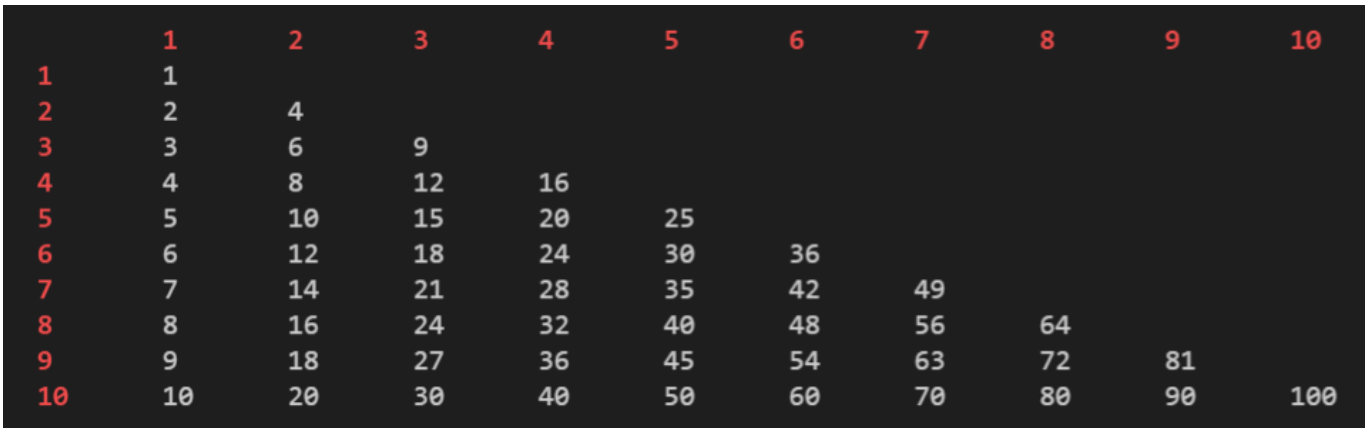
Print out a power table with 5 rows and 10 columns. Value of the power table at row i, column j is j^i. The columns should be spaced by a tab (`"\t"`).

Expected output:

```
1	2	3	4	5	6	7	8	9	10
1	4	9	16	25	36	49	64	81	100
1	8	27	64	125	216	343	512	729	1000
1	16	81	256	625	1296	2401	4096	6561	10000
1	32	243	1024	3125	7776	16807	32768	59049	100000
```

Problem 4: *Half Times Table*

A times table, or multiplication table, is a table of the products of a certain range of numbers (commonly from 1-10). Each cell of the table is the product of its row and column. The following is an image of a half of a times table from 1-10.



	1	2	3	4	5	6	7	8	9	10
1	1									
2	2	4								
3	3	6	9							
4	4	8	12	16						
5	5	10	15	20	25					
6	6	12	18	24	30	36				
7	7	14	21	28	35	42	49			
8	8	16	24	32	40	48	56	64		
9	9	18	27	36	45	54	63	72	81	
10	10	20	30	40	50	60	70	80	90	100

Figure 1: Half Times Table

For this problem, you will be displaying the half times table results (the numbers in white of the image above). However, there are a few rules:

- For every row (horizontal line), there is a 50% chance it will be displayed in reverse order. For example, for row 4, it could be displayed as `4 8 12 16`, or it could be displayed as `16 12 8 4`.
- Use **for loops** instead of while loops for this.
- You may not modify the `end` parameter of the `print` function (if you don't know what this means, don't worry about it.)

Your output will look something like this:

```
1
4     2
9     6     3
16    12    8     4
5     10    15    20    25
6     12    18    24    30    36
49    42    35    28    21    14    7
64    56    48    40    32    24    16    8
9     18    27    36    45    54    63    72    81
100   90    80    70    60    50    40    30    20    10
```

Hints:

- The bulk of your program will be in one big for loop, which will go through every row of the table.
- For each row of the table, you will be doing different things depending on a certain condition. What does that sound like out of the structures you have learned?
- For each row, you may also want to have another loop that gives you the values of the products for each cell (row * column)
- Directly printing out a product as soon as you find it is not going to work, since `print()` will add a new line. Instead, you can use *string concatenation* to concatenate each product to a string that stores all the values in a single line. Then, you can print this string after finishing one line. (If you define this string in the right place, it will reset by itself after each line.)
- You can just add a space in between each cell of the table, or if you want it to look nice, add a `"\t"` character in between each cell instead.

Problem 5: *I Think Python Really Likes Me!*

`Python` has really gotten to know you over the last few weeks and so, `Python` wants to send you a "XOX" message! It needs to know how many characters wide its message can be.

Create a program that asks for **one** input: the character-length of the message. Then the program will display an output containing `Python`'s message along with a personal signature from `Python`. X's and O's are expected to be drawn as follows:

```
X   X
 X X
  X
 X X
X   X
```

```
 000
0   0
0   0
0   0
 000
```

Note: The "O" drawing does *not* have an "O" character in the corners of the drawing. You may also notice each drawing fits within a square.

`Python` wants to make sure its message looks as pretty as possible. So, we're only concerned how the output looks in cases where the message length is at least 3 characters wide and an odd number (since the "X"s and "O"s can look a little unusual otherwise).

The following are examples of possible outputs:

```
Python needs to tell you a secret. How many characters wide can its message be? 3
X X
 X
X X
 0
0 0
 0
X X
 X
X X

- From Python
```

```
Python needs to tell you a secret. How many characters wide can its message be? 5
X   X
```

```
 X X
  X
 X X
X   X
 000
0   0
0   0
0   0
 000
X   X
 X X
  X
 X X
X   X

- From Python
```

Restriction: You may not use the `end` parameter of print nor string multiplication.

Hints:

- Break the problem up into parts: How would you print a single "X" drawing?, how would you print a single "O" drawing?
- Think of each drawing as a table where some cells contain a character (either "X" or "O") and others contain a whitespace. How can *nested for-loops* help you populate this table with the right values?
- Remember, each drawing fits within a square space. This information could help you with the "X" drawing when deciding which cells should be filled with a "X" and which should be filled with a whitespace.
- You might find printing each character right away can make formatting the overall drawing difficult. What if instead of printing character by character, we print row by row (following the table analogy from the previous hints)? How would you keep track of the current row's results so you can display it once you finish populating the row? (Think: string concatenation)