CS-UY 1114: Lab 5

looping and problem solving

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: *Just four kids*

A couple plans on having four children. Is it more likely they will have two boys and two girls, or three of one sex and one of the other?

Do this problem on a piece of paper or a document on your computer. No code is required.

Problem from Perilous Problems For Puzzle Loevers by Alex Bellos

Problem 2: *Fibbonacci*

The Fibonacci sequence is a sequence in which each number of the sequence is the sum of the previous two numbers. The first two numbers of the sequence are defined as 1 and 1. Observe for the following first ten numbers of the sequence that the sum of the previous two numbers gives the next number:

```
1, 1, 2, 3, 5, 8, 13, 21, 34, 55
```

For example, `1 + 1 = 2, 1 + 2 = 3, 2 + 3 = 5` and so on.

Create a new Python file and write code that takes a positive integer, n, and prints the first n terms of the Fibonacci sequence. For example, a sample code execution shown below is:

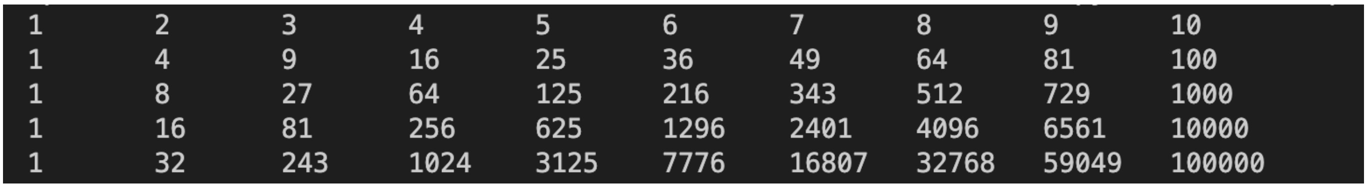
```
Please enter a number: 7
1
1
2
3
5
8
13
```

Restriction: You may not use recursion.

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. The columns should be spaced by a tab.

Expected output:

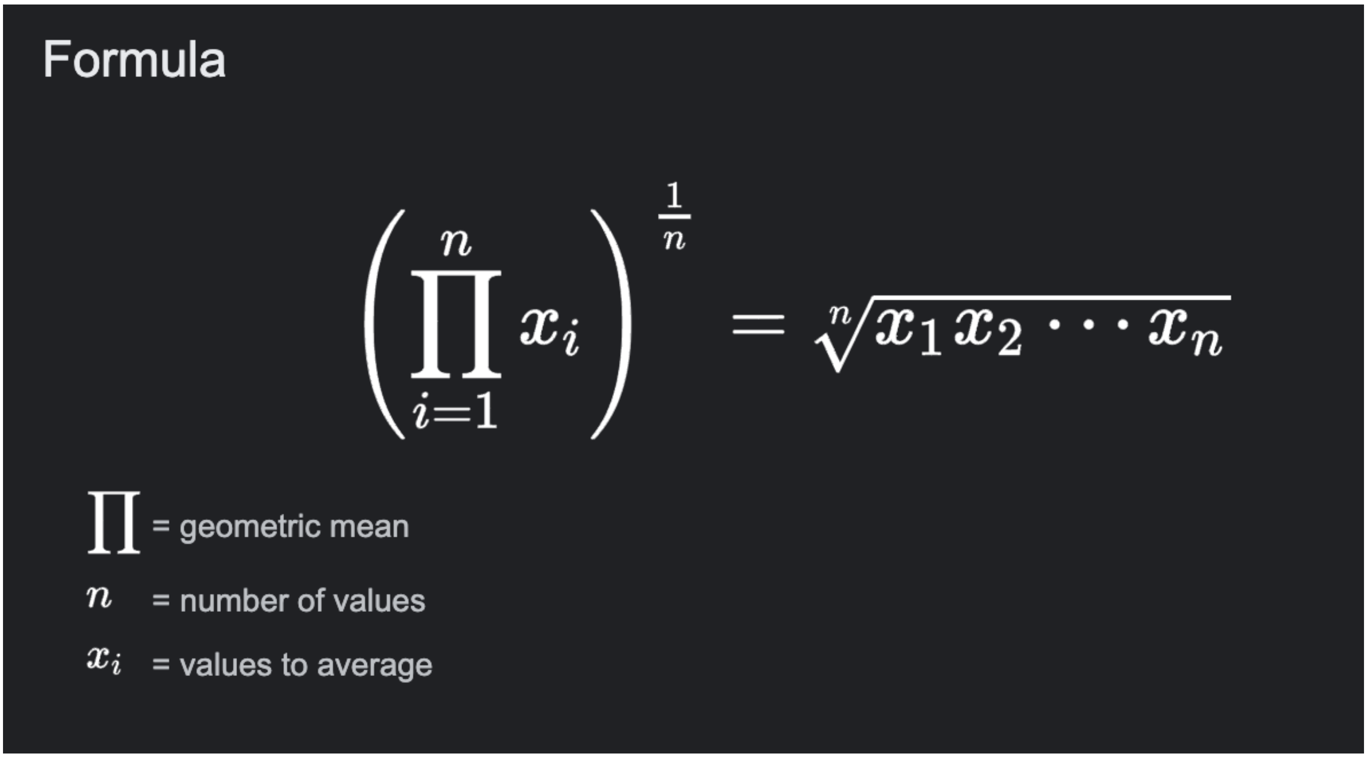


```
1       2       3       4       5       6       7       8       9       10
1       4       9       16      25      36      49      64      81      100
1       8       27      64      125     216     343     512     729     1000
1       16      81      256     625     1296    2401    4096    6561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

Problem 4: *Geometric Mean*

Write two versions of a program that reads a sequence of positive integers from the user,calculates their geometric mean,and print the geometric mean.

Note: In mathematics, the geometric mean of a data set {a_1, a_2, a_3 ... a_n} is defined by the following equation.



Formula

$$\left(\prod_{i=1}^{n} x_i\right)^{\frac{1}{n}} = \sqrt[n]{x_1 x_2 \cdots x_n}$$

$\prod$ = geometric mean

n = number of values

x_i = values to average

Your two versions of the program should read the integer sequence in two ways:

Version A:

The user will first input the length of the sequence. Sample output shown below:

```
Please enter the length of the sequence: 3
Please enter your sequence number:
1
2
3
The geometric mean is: 1.8171
```

Version B:

Read the sequence until the user enters `done`. Sample output shown below:

```
Please enter a non-empty sequence of positive integers,each one in a
separate line. End your sequence by typing done:
1
2
3
done
The geometric mean is: 1.8171
```

Hint: Think about what type of loop best suits each version of this program.

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
```

```
 OOO
O   O
O   O
O   O
 OOO
```

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
 O
O O
 O
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
 OOO
O   O
O   O
O   O
 OOO
X   X
 X X
  X
 X X
X   X

- From Python
```

Restriction: You may not use the `end` parameter of print.

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)