**ENGR 102 – Fall 2021**

**Lab Assignment #6b**

**Deliverables:**

There are three deliverables for this individual assignment. Please submit the files to Mimir:

* Lab6b\_Act1.py
* Lab6b\_Act2.py
* Lab6b\_Act3.py
* Lab6b\_Act3\_challenge.py (optional)

**Activity #1: The Collatz Conjecture**

The Collatz conjecture, also known as the conjecture (and other names), deals with the following operation to produce a sequence of numbers. Given a number, , if is even then the next number is divided by 2. If is odd, then the next number is . The Collatz conjecture is that this sequence of numbers always eventually reaches 1. As simple as this seems, it is unproven (and considered extremely hard to prove) by mathematicians.

As an example of a sequence, if you start with the number 6, then the terms of the sequence will be: 6, 3, 10, 5, 16, 8, 4, 2, 1.

Write a program named Lab6b\_Act1.py that takes in a positive integer from the user, and computes the Collatz sequence, printing out all the numbers in the sequence, followed by a line stating how many iterations it took to reach the value 1. Format your output as shown below. User input is shown in bold and red text.

Example output (using input **6**):

Enter a positive integer to compute the Collatz sequence: **6**

Here is the Collatz sequence starting at 6:

6, 3, 10, 5, 16, 8, 4, 2, 1

It took 8 iterations to reach 1

**Activity #2: Computing Sums and Products**

Write a program named Lab6b\_Act2.py to retrieve a positive integer from the keyboard and perform the two calculations below. Format your output as shown below. User input is shown in bold and red text. You **MUST** use a loop. Do **NOT** use built-in functions like sum(), etc. Do **NOT** use sympy.

1. Find the sum of all integers from 0 to the number entered.
2. Find the product of all integers from 1 to the number entered.

Example output (using input **5**):

Enter an integer: **5**

The sum of all integers from 0 to 5 is 15

The product of all integers from 1 to 5 is 120

**Activity #3: Kaprekar’s Constant**

6174 is known as Kaprekar’s Constant (<https://en.wikipedia.org/wiki/6174_(number)>) after the Indian mathematician D. R. Kaprekar. This number can be obtained using Kaprekar’s routine:

1. Take any four-digit number that has at least two different digits (add leading zeros to numbers with fewer than four digits)
2. Sort the digits in descending and then in ascending order to get two four-digit numbers, adding leading zeros if necessary
3. Subtract the smaller number from the bigger number to get a new four-digit number
4. Go back to step 2 and repeat

Kaprekar’s routine will always reach its fixed point, 6174, in at most 8 iterations. Once 6174 is reached, the process will continue to yield 7641 – 1467 = 6174.

For example, choose 3524: For example, choose 137 (add a leading zero):

5432 – 2345 = 3087 7310 – 0137 = 7173

8730 – 0378 = 8352 7731 – 1377 = 6354

8532 – 2358 = **6174** 6543 – 3456 = 3087

8730 – 0378 = 8352

8532 – 2358 = **6174**

The only four-digit numbers for which Kaprekar’s routine does not reach 6174 are repdigits (numbers where all digits are the same, such as 1111) which give 0000 after a single iteration. All other four-digit numbers eventually reach 6174, as long as leading zeros are used to keep the number of digits at four.

Write a program named Lab6b\_Act3.py that takes in an integer from the user between 0 and 9999 and implements Kaprekar’s routine. Have your program output the sequence of numbers to reach 6174 and the number of iterations to get there. Format your output as shown below.

Example output (using input **2025**):  
Enter a four-digit integer: **2025**

2025 > 4995 > 5355 > 1998 > 8082 > 8532 > 6174

2025 reaches 6174 via Kaprekar's routine in 6 iterations

Hints:

* Consider converting numbers to strings for sorting, and back to numbers for calculating
* Pad the number with 0s when it has fewer than four digits (remember string concatenation?)
* The min(), max(), and len() functions may be helpful
* Do **NOT** use any containers like lists, tuples, sets, and dictionaries

**Activity #3 challenge: More Kaprekar’s Constant (optional 5 bonus points)**

Modify your program from Activity 3 to compute the sum of the number of iterations required to reach 6174 (or 0000) using Kaprekar’s routine for all four-digit numbers, from 0000 to 9999. Name your file Lab6b\_Act3\_challenge.py. If your program calculates the total number of iterations correctly, you will receive 5 bonus points on this assignment.

Example output:  
Kaprekar's routine takes ????? total iterations for all four-digit numbers