PROGRAMMING ASSIGNMENT #2
CS 2223 D-TERM 2020
RECURRENCES, LUCAS NUMBERS, & BRUTE FORCE/EXHAUSTIVE SEARCH

SEVENTY-SIX POINTS
DUE: SATURDAY, APRIL 11, 2020 2 PM

1. The Lucas numbers, named for 19th century French mathematician Édouard Lucas, are defined exactly as the Fibonacci numbers, except that they have ever-so-slightly different initial conditions. The Lucas numbers are given by:

   $L(0) = 2$
   $L(1) = 1$
   $L(n) = L(n-1) + L(n-2)$ for $n > 1$

   Write a Java program that accepts as input a value n and writes as output the sequence $L(0), L(1), \ldots, L(n)$. Your program should compute the values recursively with a separate function that uses the definition above. Do not use an accumulator; instead, implement the recursion naïvely with the function calling itself twice. You may hardcode the initial conditions.

   (11 Points)

   What else is Lucas known for?

2. Use an available method in Java to access the system clock to investigate the order of growth of your recursive algorithm that computes the Lucas numbers.

   - Extend your program from Part 1 above to determine the time needed to compute each of the first 40 Lucas numbers. (You are encouraged to go higher!) Display these results for each n with output to your terminal/screen.
   - Have your code examine the ratio of successive calculations, i.e. $\frac{\text{Time}(L(n+1))}{\text{Time}(L(n))}$.
     Do you recognize this number?
     What is the order of growth of your algorithm?

   (18 Points)

3. The Subirachs Magic Square–pictured below–is an interesting construction. While not *technically* a magic square, its rows, columns, diagonals, corners, center, and "postage stamps" do all have the same sum. In fact, there are many other *combinations* in the square that also have this sum.

(47 Points)

- Write a Java program that counts all the 4-element combinations that have the same sum as the rows/columns, etc.

- Add to your program so that you can count all combinations with this sum. Some will have fewer than 4 elements, some will have exactly 4 elements, some will have more than 4 elements. Count them all.

- Make yet another addition to your program that counts the number of ways every possible sum can be formed. Include 0 as a possible sum; the largest sum will be created by summing every cell of the square.

- What sum can be created with the greatest number of combinations, and how many combinations is that? Notice anything interesting about that?

Bonus: The Subirachs Magic Square is fascinating. Your investigations might have piqued your curiosity about it. Write (and document!) a Java program investigating some aspect of it not covered here, *or* write a 1-page report detailing where it can be found, explaining why it is named what it is named, and discussing its "mystical" significance. (2 Bonus Points)