PROGRAMMING ASSIGNMENT #2
CS 2223 B-TERM 2019
RECURRENCES, LUCAS NUMBERS, & BRUTE FORCE/EXHAUSTIVE SEARCH

FORTY POINTS
DUE: FRIDAY, NOVEMBER 8, 2019 6 PM

1. Consider the recurrence $R(n) = 4R(n-1) - 3R(n-2)$ with initial conditions $R(0) = 2$ and $R(1) = 8$. (4 Points)

   Find its characteristic equation, and use it to create a formula for the recurrence that depends only on n, i.e. is not recursive.

   What is the recurrence's order of growth? Use $\Theta(n)$ if you can.

2. 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 C++ 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. You may hardcode the initial conditions. (7 Points)

   What else is Lucas known for?

3. Use the **clock()** function from $< time.h >$ to investigate the order of growth of your recursive algorithm that computes the Lucas numbers. (11 Points)

   - Extend your program from Part 2 above to determine the time needed to compute each of the first 30 Lucas numbers. (You are encouraged to go higher!) Display these results for each n on the standard console output.
   - 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?

4. The Suribachs 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. (18 Points)

   - Write a C++ 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 Suribachs Magic Square is fascinating. Your investigations might have piqued your curiosity about it. Write (and document!) a C++ 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)