CS 211 – Data Structures
Lab for Week #04

For this lab exercise, you will work either in 2-person teams or individually. You are HIGHLY ENCOURAGED to work in 2-person teams! The point here is to have teams discuss the lab and offer each other support in making sure the IDE of your choice is working.

Lab Exercise

It's time to try some Tail Recursion! Tail Recursion is a worthwhile goal to achieve when writing recursive code, especially when it's possible the code will go many levels deep into the recursion. But its requirement that the recursive call be absolutely last before the function returns makes it a bit tricky to write.

Download the supplied **main.cpp** and **week04Lab** files, and interactively test your functions. Write all function code in the single CPP file named **week04Lab.cpp**. Make sure the function headers match the ones in the **week04Lab.h** file!

- Write and test a function
  unsigned long long int euclideanGCD(unsigned long long int a, unsigned long long int b)
  that **uses tail recursion** and returns the Greatest Common Divisor (that is, the largest factor) of two integers from 1 to 2^64 – 1. The useful equation is (after making sure a is greater than b) is:
  euclideanGCD(a,b) = euclideanGCD(b, a % b)
  The recursion stops when you reach **b == 0**.

- Write and test the function **double geometricSum(unsigned int n)** from last week's lab that **uses tail recursion** to calculate and return the value of $1 + ½ + ¼ + ... + (½)^n$. Note that this sum can be expressed as $\sum_{i=0}^{n} (\frac{1}{2})^i$
  Do NOT use an exponent operation – instead make use of only addition and multiplication/division (with NO **while** or **for** loops!) and recursive calls.
  HINT: Consider writing the function so that all it does calls a "hidden helper" function that actually does the tail recursion! This allows the recursive function to have more arguments/parameters. This trick will be very useful later this semester as well. Remember, the recurrence relation is
  geometricSum(n) = 1 + (0.5 * geometricSum(n-1))

- Write and test a function **double smallestElement(double anArray[], int arraySize)** that returns the smallest (least) valued element. Again, NO **while** or **for** loops – use tail recursion instead! The same trick of using a hidden helper function that does the tail recursion may be useful here as well.

- At the top of your team's **week04Lab.cpp** file, include these lines:
  ```
  // CS 211 Fall 2024 – Week 04 Lab
  // <Student_Name1> and <Student_Name2>
  ```
  ONCE FINISHED, ALL STUDENTS SHOULD SUBMIT VIA CANVAS.
  You should send your CPP file to all team members for submission.
  Submit ONLY the week04Lab.cpp file via Canvas.
  Do NOT submit the main.cpp or week04Lab.h files – the instructor already has those!