# CS 1428
# Fall 2018
# Gentry Atkinson
# Lab 13

## Introduction:

You should be very familiar with functions by now. We have studied calling and defining them over the last few weeks. But let's take a minute to consider what could happen if a function calls itself (or another copy of itself). The intuitive answer is that this would cause an infinite loop of sorts. But this isn't the case if we are careful when crafting our functions. This technique is called recursion.

Every recursive function must have a "**Base Case**", which is some condition which will cause it to return a fixed value. Every call to a recursive function creates a chain of function calls which ends with the Base Case. This final function call then passes a value back up to the function instance which called it. That value is then processed and passed farther up the chain to the next instance of the function. This process continues until every instance of the function has received and returned a value. The first instance terminates last and returns its value to whatever entity called it. Be very careful when creating a recursive function to always include a base case which you know will be reached.

Recursion can be very useful and powerful or very expensive and counter-intuitive. Learning when to use recursion and when to use iteration is part of the process of learning to code.
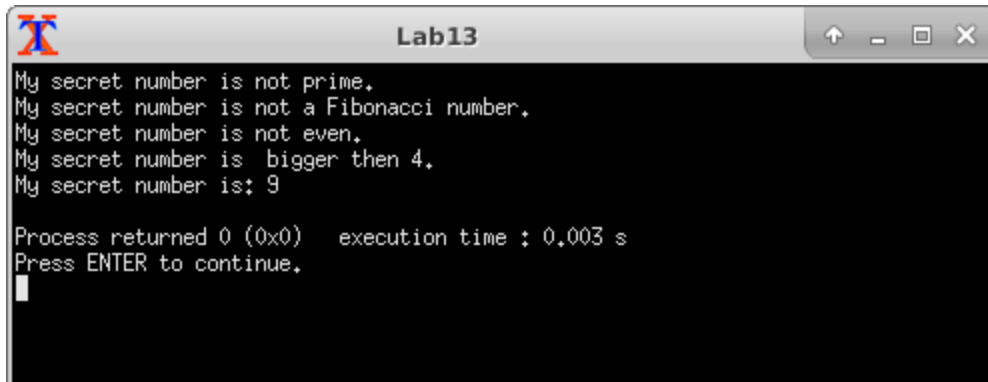
The purpose of today's lab is to introduce you to recursion and multiple file inclusions.

## Directions:

**1-** Open the command line tool by typing **cmd** into the program search bar. Use the **cd** command to change into an easy to find directory (e.g. Desktop or Documents).

**2-** Type the command **git clone https://github.com/gentry-atkinson/Lab13.git** to copy the starter project for this lab.

**3-** Launch Code::Blocks and open the Lab13.cbp project file that's in the Lab13 directory that you just copied.

**4-** Follow the directions in the comments of **main.cpp** and **Analyzer.cpp** to finish the project. Running this project will generate a secret number. Your job is complete the functions provided in the **Analyzer** class in order to guess what your secret number is. You can read the **SecretNumber.h** and **SecretNumber.cpp** files but neither will tell you your secret number.

**5-** Build and run your project. Correct any errors. Your output should look something like this:



**6-** Submit your **main.cpp**, **Analyzer.h**, and **Analyzer.cpp** file to TRACS. You can leave when you're done. Have a great winter break!