Introduction

In this week's lab, we'll give you more practice thinking about a problem BEFORE you sit down in front of a computer to write code. Remember, when planning solutions to a programming problem, it often helps to describe how you'd approach solving the problem using plain English---that is, avoid using overly technical terms like variables. *If you can't describe what you want to do in simple English, you are also likely going to have trouble coming up with the code!*

Lab Objectives

By successfully completing today's lab, you will be able to:

- Conduct a brainstorming session with one or two individuals in the class.
- Create a planning document outlining your solution to the code.
- Implement a programming project that uses input/output, conditionals, and formatted output.
- Get to know people in your 1011 lab, again!

Prior to Lab

- You should be familiar with input and output, conditionals such as if, if/else, and switch statements.
- This lab corresponds with zyBooks Chapters 3.1 – 3.10.
- The extra credit portion of this lab corresponds with zyBooks Chapters 4.1-4.4.

Deadline and Collaboration Policy

- Part 2 of this assignment is due by 11:59 PM on Tuesday (9/17/2019) via Canvas.
- Part 3 of this assignment is due by 11:59 PM on Friday (9/20/2019) via Canvas.
    - More instructions on what and how to submit are included at the end of this document.
- You should write your solutions to Part 2 and Part 3 of this lab by yourself. In this lab, you will talk at a high level with others in your lab about a solution. However, you will create all code by yourself and you should not talk about specific code to anyone but a course instructor or lab teaching assistant.
- Your zyBooks chapters and lecture slides are available resources you can use to assist you with this lab.

Lab Instructions

This week's lab involves games! We'll be implementing a solution checker for a simple game.

Sudoku

If you've never played, Sudoku is a puzzle game that uses numbers, consisting of nine squares, with nine squares within each of those squares. Within each small square, the player uses the numbers 1 through 9 so that each large square, row, and column all have exactly one instance of each number.

You can probably imagine that once you think you've found a solution, checking it would be a bit tedious, so today we'll write a program to check if the player's solution is correct. However, instead of us telling you exactly how and what you need to code, we will give you some background, and it's up to you and your team to figure out how you can implement a solution.

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

Example completed Sudoku puzzle
(https://en.wikipedia.org/wiki/Sudoku#/media/File:Sudoku_Puzzle_by_L2G-20050714_solution_standardized_layout.svg)

Sudoku is a fairly advanced numbers game, so it would be unfair to ask us to have you write a program to figure out the solution.[1] Instead, we are going to have you take a look at some output of a smaller version of a game (called a Magic Square), and then you will write a program to input and check if it's a Magic Square. What is a Magic Square? That's part of your research for this lab!

Sample Output

```
./a.out

Enter in the values: 1 2 3 4 5 6 7 8 9

You entered:
1 2 3
4 5 6
7 8 9

Analyzing…
Row 0 [1, 2, 3] does not work!
Row 1 [7, 8, 9] does not work!
Column 0 [1, 4, 7] does not work!
Column 2 [3, 6, 9] does not work!

This is not a magic square!
```

[1] Is solving Sudoku really that advanced? Yes! It's actually an incredibly advanced and unsolved problem in Computer Science and Mathematics. Why? Well, simply put, there's a lot of problems that the number of possible combinations make very hard to solve quickly, but are fairly easy to see if you have a correct answer. This is the P=NP problem, which you'll learn more about in future classes.

```
./a.out

Enter in the values: 4 9 2 3 5 7 8 1 6

You entered:
4 9 2
3 5 7
8 1 6

Analyzing…

This is a magic square!
```

Again, we structured this week's lab to have substantial planning on the first day, and you coding your solution on the second day:

Lab 04 Structure	
Tuesday	Thursday
30 minutes: work with 1-2 other individuals on Part 1. 20 minutes: work individually on Part 2. Submit results of Part 2 to Canvas by 11:59 PM.	50 minutes: work individually on Part 3. Submit to Canvas by 11:59 PM on Friday.

Part 1: Planning with a Group (First 30 minutes)
***** Do not use a computer or calculator in this part *****

Work with 1-2 other people in the room seated nearby, maybe even someone new! Examine the output in the prior section and start brainstorming ways that you could "solve" the problem---that is, what is the information you would need to have in order to write a program to accomplish the task.

Again, let's find out about your partner(s) and then break down the problem.

- Information about your partners
  - First and Last Name
  - Which is their "home" lecture section
  - Each person's favorite season
- What is the overall goal? (In other words, how do you know something is a Magic Square?)
  - What are the sub problems you need to solve to create the output above?
- What information presented in prior classes or labs is useful in solving the problems?
  - Note, you only need to know about conditionals to solve this. No repetition is expected.

Part 2: Planning on Your Own (20 minutes of Tuesday Lab)
***** It's okay to use a computer and/or calculator for this portion *****

Taking the information that was created during your brainstorm, write your own solution *in psuedocode*. How you choose to structure the planning document is of your choice, but you need to do the following:

- Review the explanation of *pseudocode* in the Safari playlist for this lab (https://learning.oreilly.com/playlists/1480ac0b-84d1-4ef6-97f4-f6936af15daf) or using the pseudocode.pdf document in Canvas.
- *Pseudocode* is NOT code. It is structured English that where you convey the key parts of your algorithm. For example, consider this *pseudocode* to calculate the average grade given 10 grades:

```
Set total to zero
Set grade counter to one

While grade counter is less than or equal to ten
    Input the next grade
    Add the grade into the total
    Add one to the grade counter
End While

Set the class average to the total divided by ten
Print the class average
```

- Thoroughly consider all the issues you need to fully create a program as specified above (such as how to make the formatting look correct, how many variables do you need, what type, etc.).
- Include the following mandatory header:
    - Your First (or preferred name) and Last Name
    - Your Lab and Lecture Section
    - Lab 04 – Part 2
    - Today's Date
    - Collaboration Statement: In this statement you indicate who you worked with, and which lecture sections they are enrolled in.

Submit this part of the assignment to Canvas by 11:59PM (as specified in the submission Guidelines).

Part 3: Implementation (50 minutes on Thursday).

Lab Exercise

Using your planning document from Part 2, you will create a C program called lab04a.c that creates and analyzes the Magic Square game.

Your Magic Square program should clearly display whether the input is a magic square, and clearly give any reasons why input is not a Magic Square. You should use the same phrasing as in the example code above.

For this part, you should not use loops/repetition.

Remember to include the following information in the header of each C source code file that you submit for grading.

```
/*
* filename.c
* Author: Your Name
* Lecture, Lab Section: Insert Lecture Section #, Insert Lab Section #
* Lab #: Insert Lab #
* Submitted on: Insert Date
*
* Purpose: 1 or 2 sentences stating what the program does.
*
* Academic Honesty Declaration:
* The following code represents my own work and I have neither received nor given assistance that violates the collaboration policy posted with this assignment. I have not copied or modified code from any other source other than the lab assignment, course textbook, or course lecture slides. Any unauthorized collaboration or use of materials not permitted will be subjected to academic integrity policies of Clemson University and CPSC 1010/1011.
*
* I acknowledge that this lab assignment is based upon an assignment created by Clemson University and that any publishing or posting of this code is prohibited unless I receive written permission from Clemson University.
*/
```

Bonus Exercise

Create a copy of your lab04a.c file called lab04b.c.

Modify and extend your program so that it uses loops to perform the analysis. Note, your lecture section may not have covered loops yet, so if you have time to work on this, you may want to look ahead to zyBooks Chapters 4.1 – 4.4.

Submission Guidelines

- Part 1: No submission.
- Part 2: Submit your writeup to the Canvas assignment associated for this lab by 11:59PM on Tuesday (9/17/2019). You should check within Canvas to make sure your submission was uploaded correctly and in its entirety.
- Part 3: Submit your lab04a.c program to Canvas assignment associated for this lab by 11:59 PM on Friday (9/20/2019). You should verify within Canvas to make sure your submission was uploaded correctly and in its entirety.
    - If you opt to complete lab04b.c, submit the program to the associated Canvas assignment by 11:59 PM on Friday (9/20/2019). Please submit both lab04a.c and lab04b.c to the assignment if you choose to complete the optional bonus.

Grading Rubric

- If you are not present and attending lab during Part 1 and Part 2 of this lab on Tuesday, you will not receive credit for the planning portion of this lab.
- If your document for Part 2 is not submitted successfully to Canvas by the due date, you will not receive credit for the planning portion of this lab.
- If your code for Part 3 is not submitted successfully to Canvas by the due date, you will not receive credit for the coding portion of this lab.
- Your assignment will be graded out of 100 points. The approximate grading distribution is:
    - Brainstorming document completeness/accurate — 40 points
    - Program lab04a.c
        - Program operates correctly — 20 points
        - Output is formatted correctly — 20 points
    - Program compiles without errors or warnings — 10 points
    - Proper code formatting and commenting — 10 points
    - Optional bonus exercise lab04b.c — up to + 10 bonus points
        - Requires lab04a.c is submitted

Additional Resources

- You can read about Magic Squares on the web if you need to, but try to work from the examples.