CS 159 – Spring 2021 – Lab #2

What will you submit? A single C-file will be submitted electronically via the guru server. An example submission was conducted as part of the Account Configuration Activity. If you have a concern regarding how to submit work, please contact course staff prior to the deadline for this, and all, assignments. The programming assignment is due on Friday February 5, 2021 at 11:00pm (LOCAL WEST LAFAYETTE, IN TIME). **No late work will be accepted.**

Weekly Quiz #2:

The weekly quiz will be available (Week 3 module on Brightspace) until the same date and time that the programming assignment is due. It is strongly recommended that you complete the attached problems, the programming assignment, and watch all relevant lectures before attempting the quiz.

The quiz will emphasize chapter 2 material, the written problems in this document, the lab programming assignment, and the course programming and documentation standards as used in this lab. Quiz questions are presented one at a time and cannot be revisited. Be sure to save your answers to each question and to finish your quiz to ensure it is submitted for grading. Most problems on lab quizzes will be multiple-choice or true-false. Each quiz has a 15-minute time limit.

Collaborative Teaming:

- **How do I know who is on my lab team?**
  - **On-campus students:** TAs have contacted their students via e-mail.
  - **On-line students:** Visit the Start Here module on Brightspace and locate the Distance Learning Team Assignment spreadsheet.
- **What if a partner does not respond to your communication?** Then the remaining active partners need to be prepared to proceed on the assignment to meet the deadline.
- **Groups are expected to communicate to share their ideas when it comes to solving the conceptual and programming problems associated with this lab.** You may find a collaborative document to be a helpful way to share thoughts on the written problems and to formulate the logic for the programming problem. Other on-line tools may come in handy when trying to collaborate on specific segments of code, just make sure they protect your code from being posted publicly! One popular service from previous semesters is codeshare.io.
- **As a group you must determine who will make the final submission for your group**, when that submission will be made, and how the concerns regarding submission will be communicated with the other members. **Only one person per group will make submissions for the entire group**. The grader for your section cannot be expected to grade submissions from multiple members of the same group to determine which submission you actually want graded.
  - In order for each member of the group to get credit for the programming problem associated with this lab, their **Purdue University e-mail address must appear in the assignment header**.
- **How might collaboration be useful on this particular programming assignment?** First, commit to a time to collaborate on the written problems. This will strengthen the understanding of current material of all group members which will make it more likely that each member can contribute to the final effort of the programming assignment. Next, proceed with the math necessary to solve this problem and determine the correct width and precision modifiers to match the expected output **exactly.** And finally, why not share a screen while the coding is taking place? Doing such will serve to improve the code writing skills and familiarity with course tools that most students need, including a little confidence building, as we move forward this semester!

Solve the following problems related to material found in Chapter 2 and the course standards.

Statement	**True / False**
A C program begins with a section for preprocessor directives.	
The preprocessor is a part of the compiling process and prepares your code for the remainder of that process.	
The declaration section contains executable instructions for the computer.	
Every program must have exactly one function named `main`.	
The `main` function is the starting point for execution of the program.	
The `gcc` compiler as found on `guru` will require that the `main` be an `int` (integer) function.	
The `gcc` compiler as found on `guru` will require the source code file name to end with a `.c` extension.	
Within each function the local declarations and executable statements must NOT be permitted to overlap.	
All code found in the executable statement and local declaration sections of the `main` function is considered to be the body of the function.	
Variable declarations will NEVER be permitted in the global section this semester.	
The files `stdio.h` and `math.h` are libraries that contain standard functions for our use.	
The `return(0);` statement will be the final statement in the `main` function.	
The `return` statement in `main` will return control to the operating system and terminate the current program.	
Comments are added to a program to improve its level of documentation intended for other programmers.	
Section 2.5 – C programming text	
Variables are named memory locations that have a type.	
Each variable in a C program must be declared and defined before it can be used.	
It is much easier to find and work with variables if they are defined on separate lines.	
Variables in the C language are initialized automatically when they are defined.	
Section 2.6 – C programming text	
The backslash (\) is known as an escape character.	
A character constant is enclosed in double quotes.	
A literal constant is an unnamed constant used to specify data.	
The command to create a defined constant is usually placed at the beginning of the program.	
While variables have data types, constants (both literal and defined) do not.	
Section 2.7 – C programming text	
The conversion code selected for a placeholder (conversion specification) depends on the data type of the value to be displayed in a `printf` statement.	
The size modifier is used to specify the minimum number of positions to reserve for the output of a value.	
A precision modifier may be used for the output of any numeric value.	
The width modifier for the output of a value must be large enough to account for the integral value of the number, the decimal point, and the number of digits in the decimal position.	
The address list of a `scanf` function must specify where in the memory of the computer the input should be stored.	

- **Solve problem #23 from page 87 of your C programming text.** Every row provided in the table below represents one row of output produced. Each column represents enough room to display a single character. Not all rows and columns will necessarily be used.

- **Solve problem #32 from page 89 of your C programming text.**
  - Include the print statement here:
  - Does the width modifier need to account for the decimal point in the floating-point value?
  - If the precision modifier can be used to specify the number of digits to display to the right of the decimal point, then how can the width modifier be used to limit the number of digits to display to the left of the decimal point?
- **Solve problem #38 from page 90 of your C programming text.**
  - Demonstrate to your lab instructor that you have been able to write the code necessary to reproduce the output as shown in the book.
  - What does the integer represent in the second line of output generated?
  - What is the message generated by the compiler as a result of the third print statement? What does this mean?

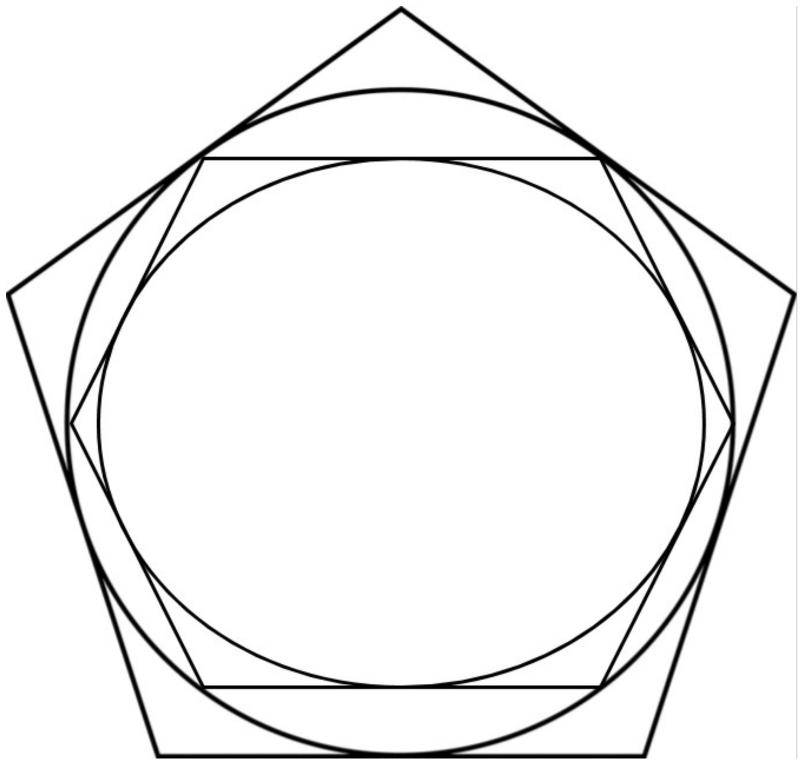
Lab #2 – Programming Assignment
Due: Friday February 5, 2021 at 11:00pm (time local to West Lafayette, IN)
10 Points Possible

Problem: In the image given there is a smaller circle inscribed inside of a regular hexagon. The hexagon is inscribed inside of a larger circle. And finally the larger circle is inscribed inside of a regular pentagon.

Given the radius of the smaller circle, calculate the following:

1. Area of the small circle.
2. Area and side length of the hexagon.
3. Area of the large circle.
4. Area and side length of the pentagon.

Accept input and display ALL output exactly as seen in the example executions that follow.



Example Execution #1:

```
Enter the radius of the small circle -> 10

-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
Area of small circle:        314.16
Hexagon side length:          11.55
Area of hexagon:             346.41
Area of large circle:        418.88
Pentagon side length:         16.78
Area of pentagon:            484.36
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
```

Example Execution #2:

```
Enter the radius of the small circle -> 17.35

-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
Area of small circle:        945.69
Hexagon side length:          20.03
Area of hexagon:            1042.77
Area of large circle:       1260.92
Pentagon side length:         29.11
Area of pentagon:           1458.04
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
```

All course programming and documentation standards are in effect for this and each assignment this semester. Please review this document!

Example Execution #3:

```
Enter the radius of the small circle -> 55.79

-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
Area of small circle:       9778.28
Hexagon side length:          64.42
Area of hexagon:           10782.10
Area of large circle:      13037.71
Pentagon side length:         93.61
Area of pentagon:          15075.87
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
```

Example Execution #4:

```
Enter the radius of the small circle -> 121.21

-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
Area of small circle:      46155.85
Hexagon side length:         139.96
Area of hexagon:           50894.11
Area of large circle:      61541.14
Pentagon side length:        203.38
Area of pentagon:          71161.76
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
```

Additional Requirements:

1. Add the **lab assignment header** (`vi` shortcut `hlb` while in command mode) to the top of your program. An appropriate description of your program must be included in the assignment header.

Include the Purdue University e-mail addresses of **each contributing group member** in the assignment header!

2. **Each of the example executions provided for your reference represents a single execution of the program.** Your program must accept input and produce output **exactly** as demonstrated in the example executions, do not add any "bonus" features not demonstrated in the example executions. Your program will be tested with the data seen in the example executions and an unknown number of additional tests making use of **reasonable data.**
   - All floating-point variables must be of the `doubele` type.
   - Use the constant value `M_PI` where needed for the constant pi.

3. Course standards **prohibit** the use of programming concepts beyond the material found in the first three chapters of the book, notes, and lectures to be acceptable for use.
   - The use of the `math.h` library is expected for access to the `sqrt`, `pow`, and trigonometric functions.

4. A program **MUST** compile, be submitted through the `guru` server prior to the posted due date to be considered for partial credit. The C-file you submit must be named exactly: `lab02.c`

Course Programming and Documentation Standards Reminders:

- Indent all code found within the `main` function **exactly** two spaces.
- Place a **single space** between all operators and operands.
- Comment **all** variables to the right of each declaration. Declare only one variable per line.
- Notice that several programs (see program 2-9 on pages 74-75) in the programming text use a single line comment to indicate the start of the local declaration and executable statement sections of the `main` function.
  - At no point during the semester should these two sections ever overlap. You might consider adopting this habit of commenting the start of each section to help you avoid this mistake.
- Select **meaningful identifiers** (names) for all variables in your program.
- Do not single (or double) space the entire program, **use blank lines when appropriate**.
- Maximize your use of symbolic/defined constants and minimize your use of literal constants.

Auto-Grade Tool

- We have implemented what is being referred to as the auto-grade tool. At the time of a successful assignment submission you may receive some feedback on your program in regards to course programming and documentation standards. This feedback may include a potential deduction that you will receive once your assignment is reviewed by your grader.
- It is expected that graders verify those notes identified by this tool to ensure that they are indeed applicable and reasonable to the submission. Graders may make additional deductions for those standards not identified by the new tool.
- We hope that this feedback helps with the enforcement of course standards, consistency in grading across sections, and to encourage students to revise their work when problems are identified before the assignment deadline passes. It is possible to resubmit an assignment for grading up to the advertised deadline. Only the final successful submission is retained and evaluated.