CS 159 – Spring 2022 – Lab #2

Overview of Current Lab:

1. **Important** – submit your attendance at the start of every lab meeting: https://www.cs.purdue.edu/cs159/lab/submit_attendance.php
2. Please read over the collaborative teaming expectations.
3. **Work with your team on the written problems found in this document.** These problems are the basis for the lab quiz and may include content not yet introduced in lecture.
4. Begin your work on the programming assignment.
5. Complete the Collaborative Group Communication form and submit to your lab instructor for review.
6. During the final 10 minutes of your lab today you will complete the second lab quiz.

Collaborative Teaming:

- **Why utilize and rotate roles among the members of a team?**
  - The use of, and rotation through, the roles described will allow each member to gain experience with the entire process of developing a solution to a programming problem. As a team there should be a shared interest in strengthening each individual member such that they are able to continue to contribute as problems become longer and increase in complexity. Individuals should insist on a rotation of roles during the entire development process to better prepare for homework programming assignments which are individual efforts.
  - The roles are designed to provide the opportunity for individual members to learn about the problem-solving process and the implementation of a solution using the tools of the course. The roles do not emphasize efficiency as each lab programming assignment is due approximately one week after it becomes available. Do not allow your desire to complete every assignment quickly be at the expense of learning the material.
- **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.
- **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.
- **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.

Lab Quiz #2 – 5 points possible: The lab quiz will be made available on Brightspace (Week #3 module) during the final 10 minutes of your lab today. The quiz will emphasize material from chapter 2 and the course programming and documentation standards relevant to the lab programming assignment. **Lab quizzes are individual efforts and you may not use any resources while completing the 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.

- Quizzes are password protected and will be provided by your lab instructor.
- The time at which you take your quiz is monitored and students completing their quiz at an unexpected time are subject to loss of points and a potential academic integrity inquiry.

Lab Programming Assignment #2 – 5 points possible: Every lab programming assignment is due 30 minutes prior to the start of your next lab meeting.

- **How might collaboration be useful on this particular programming assignment?** Each checkpoint provided outlines a process that may be beneficial in your approach to solving this problem. A little research into the math needed to solve this problem is necessary, you may find on-line calculators to be beneficial to verify the correctness of the output your program is generating.

CS 159 – Collaborative Group Communication

Name	Purdue career account name	Who will make the final submission of this lab assignment?
		The first individual listed here is responsible for the final submission of this lab programming assignment.
		These partners must be responsible for submitting at least one future lab assignment for this team.

Each member should initialize affirming the following:

Every member has the contact information of the other lab partners.			
The collaborative team has been created and finalized in Vocareum.			
How will members of the group be contacted should a concern arise with the assignment? Be specific (text message, e-mail, groupme, slack).			
When and where is the group meeting next?			
When will the final electronic submission be made?			
Who is responsible for bringing the C programming text and class notes to the next lab meeting?			

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.	T
The preprocessor is a part of the compiling process and prepares your code for the remainder of that process.	T
The declaration section contains executable instructions for the computer.	F
Every program must have exactly one function named `main`.	T
The `main` function is the starting point for execution of the program.	T
The `gcc` compiler as found on Vocareum will issue a warning when the `main` is not an `int` (integer) function.	T
The `gcc` compiler as found on Vocareum will require the source code file name to end with a `.c` extension.	T
Within each function the local declarations and executable statements must NOT be permitted to overlap.	T
All code found in the executable statement and local declaration sections of the `main` function is considered to be the body of the function.	T
Variable declarations will NEVER be permitted in the global section this semester.	T
The files `stdio.h` and `math.h` are libraries that contain standard functions for our use.	T
The `return(0);` statement will be the final statement in the `main` function.	T
The `return` statement in `main` will return control to the operating system and terminate the current program.	T
Comments are added to a program to improve its level of documentation intended for other programmers.	T
Section 2.4 – C programming text	
A `char` is any value that can be represented in the character set of the computer.	T
The character data type is one of three integral types that cannot contain a fraction part and are whole numbers.	T
On the ASCII table all of the lowercase letters are grouped together. The same is true for the uppercase characters and digits.	T
The first 32 characters on the ASCII table are control characters and do not represent alphanumeric, digit, or punctuation characters.	T
Section 2.5 – C programming text	
Variables are named memory locations that have a type.	T
Each variable in a C program must be declared and defined before it can be used.	T
It is much easier to find and work with variables if they are defined on separate lines.	T
Variables in the C language are initialized automatically when they are defined.	F
Section 2.6 – C programming text	
The backslash (\) is known as an escape character.	T
A character constant is enclosed in double quotes.	F
A literal constant is an unnamed constant used to specify data.	T
The command to create a defined constant is usually placed at the beginning of the program.	T
While variables have data types, constants (both literal and defined) do not.	F

- **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?

STOP!		**TA Verification:**
Complete Written Problems	Problems from pages 3-4 in this document are complete as the group prepares for programming assignment and quiz at the end of the lab session.	

Lab #2 - Programming Assignment
Due: 30 minutes prior to the start of your next lab meeting.
5 Points Possible

Collaborative Roles for the Lab Session

Collaborative Teaming. For this lab you will be working in your assigned teams. If you are unable to complete your assignment during the lab then it is expected that your team meet and collaborate outside of class to finish and submit the problem assigned. **Be sure to exchange contact information during lab today!**

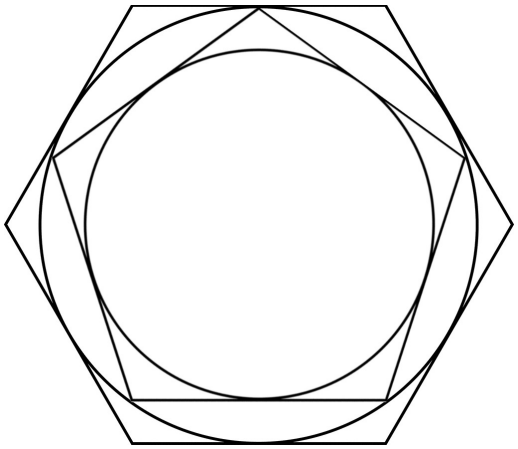
Role:	**Description: Every member will rotate roles at every checkpoint.**
Driver	The driver is in charge of the computer which includes **entering code, saving, testing, and submitting**. This individual should be soliciting the other two members for advice.
Navigator	The role of the navigator is to look over the shoulder of the driver for **syntax errors, logical errors, and concerns related to course standards**. The most common mistakes include failing to pair quotes, misplaced semi-colons, and improper placement of parentheses.
Manager	The manager may not be as close to the driver as the navigator but still plays an important role ensuring that the **algorithm to be implemented is correct** and can be tested using a variety of input to verify correctness.

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

Given the side length of the hexagon, calculate the following:

- Area of the hexagon.
- Radius and area of the large circle.
- Apothem, base, and area of the pentagon.
- Area of the small circle.

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



Checkpoints During Lab #2:		**TA Verification:**
1 – Plan the Mathematics	Identify all variables (plus data types) and expressions (and/or functions from `math.h`) needed to solve this problem.	
2 – Getting Started in Vocareum	File `lb02.c` created, assignment header completed, `main` function inserted, variables declared and commented.	
2 – Input and Calculate	Prompts for user input, accepting user input, and necessary calculations.	
3 – Formatted Output	Display all output **EXACTLY** as seen in the example executions.	
4 – Successful Submission	Save, compile, and test program. Submit and review report related to expected output and course standards.	

Example Execution #1:

```
Enter the hexagon side length -> 10

-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
Area of hexagon:           259.81
Radius of large circle:      8.66
Area of large circle:      235.62
Apothem of pentagon:         7.01
Base of pentagon:           10.18
Area of pentagon:          178.32
Area of small circle:      154.21
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
```

Example Execution #2:

```
Enter the hexagon side length -> 10.5

-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
Area of hexagon:           286.44
Radius of large circle:      9.09
Area of large circle:      259.77
Apothem of pentagon:         7.36
Base of pentagon:           10.69
Area of pentagon:          196.60
Area of small circle:      170.02
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
```

Example Execution #3:

```
Enter the hexagon side length -> 9.25

-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
Area of hexagon:           222.30
Radius of large circle:      8.01
Area of large circle:      201.60
Apothem of pentagon:         6.48
Base of pentagon:            9.42
Area of pentagon:          152.58
Area of small circle:      131.95
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
```

Example Execution #4:

```
Enter the hexagon side length -> 27.75

-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
Area of hexagon:          2000.68
Radius of large circle:     24.03
Area of large circle:     1814.42
Apothem of pentagon:        19.44
Base of pentagon:           28.25
Area of pentagon:         1373.20
Area of small circle:     1187.55
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
```

Additional Requirements:

1. Add the **lab assignment header** (`vi` shortcut `:hlb` while in command mode) to the top of your program. An appropriate description the logic of your program must be included 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 `double` type.
    - Use the constant value `M_PI` where needed for the constant pi.
3. Course standards **prohibit** the use of programming concepts not yet introduced in lecture. For this assignment you can consider all material 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 Vocareum as demonstrated during the lab #0 exercise, and submitted prior to the posted due date to be considered for credit. The C-file you submit must be named exactly: `lb02.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.