CS 159 – Spring 2021 – Lab #1

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 January 29, 2021 at 11:00pm (LOCAL WEST LAFAYETTE, IN TIME). **No late work will be accepted.**

Weekly Quiz #1:

The weekly quiz will be available 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 1 & 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:** A course staff member will e-mail you by Monday January 25th and include the contact information of your lab partners.
  - **On-line students:** Visit the Start Here module on Brightspace and locate the Distance Learning Team Assignment spreadsheet on Monday January 25th.
  - **All students:** You should begin with an e-mail to your partners and introduce yourself. It is nice to share with your partners your major, your plans for the semester, and a little about your previous programming experiences.
- **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?** This assignment is not particularly long or logically challenging but it will test your understanding of input, formatting output, and applying the course standards. Each member may consider solving this problem individually and then sharing their work **with the group** so that the best aspects from among the efforts can be selected and modified to maximize points earned and understanding of the programming concepts applied.

Solve the following problems related to material found in Chapter 1, Chapter 2, and the course standards.

Statement	True / False
Section 1.1	
The CPU is responsible for executing instructions for the computer.	F
Main memory is a place where the programs and data are stored temporarily during processing.	T
Secondary storage, similar to main memory, also stores programs and data.	T
Section 1.3	
Each computer has its own machine language which is made of streams of 0's and 1's.	T
Symbolic languages use mnemonics to represent the various machine language instructions.	T
Only the machine and symbolic languages can be understood by computer hardware.	T
One benefit to the use of symbolic languages is improved programming efficiency compared to a machine language.	T
High-level languages are portable to many different computers.	T
The process of converting a high-level language to machine language is known as compilation.	T
Section 1.4	
The software used to write programs is known as a text editor.	T
A source file contains code written in a programming language that is to be sent to the compiler.	T
One purpose of preprocessor commands is to tell the preprocessor to make substitutions in the code.	T
The preprocessor prepares the code for translation into machine language.	T
Section 1.5	
Careful design of software can result in programs that will be efficient, error-free, and easy to maintain.	T
Writing code is the first step in the waterfall model of developing software.	F
The waterfall model of developing software places an emphasis on planning and testing.	T
Pseudo-code and flowcharts can be used to represent individual algorithms of a program.	T
A structure chart can be used to represent the entire program.	T
Pseudo-code uses exact programming language syntax to represent a module in the larger program.	F

A flowchart uses geometric symbols to represent the logic of a function, or module, in the larger program.	T
Except for the most simple program, one set of test data will not completely validate a program.	T
Section 1.6	
Software engineering is the use of sound engineering methods and principles to obtain software that is reliable.	T
The three constructs of structured programming are: (1) sequence, (2) selection, and (3) looping.	T
Section 2.1	
A high-level language allows the programmer to concentrate on the problem at hand and not worry about the specific machine that the program will be using.	T
The C programming language is considered a structured programming language.	T
The C programming language is not considered a high-level programming language because of its age.	F
The following section is from course programming and documentation standards.	
All code found between { and } should be indented two additional spaces.	T
Place a single space between all operators and operands.	T
Comment all variables to the right of each declaration.	T
Declare only one variable per line.	T
Select meaningful identifiers (names) for all variables in your program.	T
Do not single (or double) space the entire program, use blank lines when appropriate.	T
Output of your final `lab01.c` submission must match **EXACTLY** that seen in the example executions.	T
There is no need to include example output with your submission.	T

The following questions are in regards to Program 2-7 found on pages 71 and 72 of your C programming text:

What is the purpose of the statements on lines 12 through 14?

The purpose is to declare that the variables should all be converted to the float form.

How are the statements on lines 17 and 18 related in this program?

line 17 tells the compiler to print out the text.
line 18 tells the compiler to accept the input the user entered after the execution of line 17.

Can the statements on lines 20 and 21 be moved before those on lines 17 and 18? Why or why not?

No because the radius is known after line 17 & 18 so if we move it before them, it wouldn't know what radius is and therefore will have an error.

Can the statements on lines 20 and 21 be moved after those on lines 23 through 25? Why or why not?

No because line 25-25 prints the answers gotten from line 20-21.

On line 23, what are the `\n` and `%10.2f` **inside** of the quotes accomplishing for the print statement?

The \n stands for start a new line. the % 0.2 shows that 2 characters should show after the decimal pt. f stands for the data type, which is float.

On line 25, explain why an additional `\n` should be placed inside of the quotes.

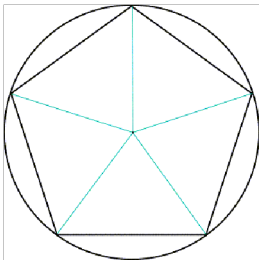
An addition \n should be placed inside of the quotes because the \n at the start Is used to make sure this line starts a new line below the previous line and we should put another \n so that we make sure whatever'

Lab #1 – Programming Assignment

Due: Friday January 29, 2021 at 11:00pm (time local to West Lafayette, IN)

10 Points Possible

Problem: Given the radius of a circle, calculate the **area of that circle** along with the **length of a side** and **area of a regular pentagon** inscribed in the circle **using the formulas and process described below.** Use the constant and value of PI from program 2-7 found on pages 71 and 72 of your C programming text. Assume all numeric values are of the `float` data type. **Accept input and display ALL output exactly as seen in the example executions that follow.**

$sidelengthof\ pentagon = 2 \times radius \times \sin(\frac{\pi}{5})$	Now that you know the length of the radius, can calculate the length of the side of the pentagon, you can proceed with calculating the area of one of triangles using the Pythagorean Theorem. The sum of the areas of the five triangles equals the sum of the area of the inscribed regular pentagon. 

Example Execution #1:

```
Please enter the value of the radius -> 27

Circle Area: 2290.23
Pentagon side: 31.74
Pentagon area: 1733.30
```

Example Execution #2:

```
Please enter the value of the radius -> 15.4

Circle Area: 745.06
Pentagon side: 18.10
Pentagon area: 563.88
```

Example Execution #3:

```
Please enter the value of the radius -> 8.33

Circle Area: 217.99
Pentagon side: 9.79
Pentagon area: 164.98
```

Example Execution #4:

```
Please enter the value of the radius -> 15.825

Circle Area: 786.75
Pentagon side: 18.60
Pentagon area: 595.43
```

Example Execution #5:

```
Please enter the value of the radius -> 0

Circle Area: 0.00
Pentagon side: 0.00
Pentagon area: 0.00
```

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

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 `float` type.
   - Use the value `3.1416` where needed for the constant pi similar to how it is defined on page 71 of the C programming text.
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: `lab01.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**.
- There is no need to include example output with your submission. **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.