CS 159 – Spring 2021 – Lab #3

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 12, 2021 at 11:00pm (LOCAL WEST LAFAYETTE, IN TIME). **No late work will be accepted.**

Weekly Quiz #3:

The weekly quiz will be available (Week 4 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 3 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?**
  - Consider the logic of the problem and the approach of implementing calculations to resemble selection: Is the radius of the circle the apothem of the polygon as input by the user or does it need to be calculated in another way? There are two options that correspond with the two options available as the first input to the program. What expressions can be developed to activate one method for calculating the radius while canceling out the other?
  - Refine your process of collaboration: As programs increase in complexity and length it will become imperative that groups establish expectations of timely communication and that all partners remain current in course material to be meaningful contributors to lab assignments.

Solve the following problems related to material found in Chapter 3 and the course standards.

Statement	True / False
The coding assignment for lab #3 forbids the use of selection constructs (if, else if, else, switch), logical operators (AND, OR, NOT), and relational operators (<, <=, >, >=, ==, !=).	TRUE
All of the compound assignment operators share the same level of operator precedence.	F
Operator precedence can be used to bind operators to operands and from that will determine the order of operations in an expression.	T
A single-type operation will generate a result of that same type.	T
When evaluating an operator with mixed-type operands it is the lower ranked data type that is converted to that of the higher ranked data type.	T
The use of a precision modifier when displaying a floating-point value will result in truncating all digits beyond the specified precision value.	T
Converting a higher ranked data type to that of a lower ranked data type may result in the loss of data.	T
Section 3.1	
When two operators with the same precedence occur in an expression and their associativity is left-to-right, the left operator is evaluated first.	T
An expression always reduces to a single value.	F
The value of the postfix increment expression is determined before the variable is increased.	T
The operand in a postfix or prefix expression must be a variable.	T
The assignment expression evaluates the operand on the right side of the operator and places its value in the variable on the left.	T
When a compound assignment is used with an expression, the expression is only evaluated first when parentheses are used to raise its level of precedence. [Example: `x *= (a + b)`]	T
The compound assignment operator (/=) has a higher level of precedence then the addition operator (+).	F
Section 3.4	
The result of an expression is undefined when it attempts to modify a single variable more than once.	T
Section 3.5	
When the types of two operands in a binary expression are different, C automatically converts one type to the other.	T
When the types of two operands in a binary expression are different, the lower-ranked type is promoted to the rank of the higher type.	T
In an assignment statement, promotion occurs if the right expression has a lower rank than the variable on the left and demotion occurs if the right expression has a higher rank.	F
An explicit type conversion is the programmer taking control and determining the data type of an operand in an expression.	T
To cast data from one type to another, we specify the new type in parentheses before the value we want converted.	T
Section 3.6	
The defined constant results in an automatic substitution of the value that follows the symbol where it is found in the program. One exception, no such substitution will take place inside of quotes.	F
Problems associated with defined constants are difficult to resolve as the programmer views the original statement and not the statement with the error after the substitution takes place.	T
Section 3.7	
One programming technique to simplify code is to use parentheses, even when unnecessary.	T

Solve the following problems related to standard library constants and functions:

In which library will you find the `abs` function? How is this function different from the `fabs` function found in `math.h`?

<stdlib.h> abs() is for integer values and fabs() is for floating type data.

What is `INT_MAX`? Where is it defined? What are `FLT_MAX` and `FLT_DIG`? Where are these defined?

<limits.h> INT_MAX is a macro that specifies that an integer variable cannot store any value beyond this limit.

<float.h> FLT_MAX define the max finite floating point value. FLT_DIG define the max # decimal digits

Identify each of the constants found `math.h` below:

`M_E`

The base of natural logarithms

`M_LOG10E`

The logarithm to base 10 of M_E

`M_PI`

Pi: 3.141....

`M_LN2`

The natural logarithm of 2

`M_LOG2E`

The logarithm to base 2 of M_E

`M_LN10`

The natural logarithm of 10

Solve the following problems related to material found in Chapter 3.

- 25 % 11

  3

- 132 % 2

  0

- 134 % 2

  0

- 16 % 17

  16

- 49 % 5

  4

- 133 % 2

  1

- 135 % 2

  1

- 2 % 1

  0

Given A and B are integer variables, both are greater than zero, and A is greater than B. What is the result of the expression below?

- B % A

  B

Given A and B are integer variables, both are greater than zero, and A is less than B. What is the range of values possible as a result of the expression below?

- B % A

  from 0 to B-1

Lab #3 – Programming Assignment
Due: Friday February 12, 2021 at 11:00pm (time local to West Lafayette, IN)
10 Points Possible

Problem: The user of your program will indicate whether (1) a regular polygon is inscribed inside of a circle or (2) the circle is inscribed inside of a regular polygon. The number of sides and the apothem will follow as additional input. Calculate and display both the radius and area of the circle, and the side length and area of the regular polygon.

Requirements of your solution:

- The **only functions** from `math.h` that can be used are `sqrt`, `pow`, and the trigonometric functions.
- Any use of logical operators, relational operators, `bool` variables, or selection constructs is **prohibited**.
- A violation of either of the above will result in your assignment being awarded no points.

The technique necessary to solve this problem with the given requirements is similar to that used in the final two examples of chapter 3.

Example Execution #1:

```
1. Polygon inside of circle
2. Circle inside of polygon
Select your option -> 1
Enter number of polygon sides -> 6
Enter length of polygon apothem -> 9

-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
Circle Radius:            10.39
Circle Area:             339.29
Polygon Side Length:      10.39
Polygon Area:            280.59
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
```

Example Execution #2:

```
1. Polygon inside of circle
2. Circle inside of polygon
Select your option -> 2
Enter number of polygon sides -> 6
Enter length of polygon apothem -> 20

-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
Circle Radius:            20.00
Circle Area:            1256.64
Polygon Side Length:      23.09
Polygon Area:           1385.64
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
```

Example Execution #3:

```
1. Polygon inside of circle
2. Circle inside of polygon
Select your option -> 2
Enter number of polygon sides -> 5
Enter length of polygon apothem -> 138.83

-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
Circle Radius:           138.83
Circle Area:           60550.33
Polygon Side Length:     201.73
Polygon Area:          70016.06
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
```

Example Execution #4:

```
1. Polygon inside of circle
2. Circle inside of polygon
Select your option -> 1
Enter number of polygon sides -> 5
Enter length of polygon apothem -> 55.71

-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
Circle Radius:            68.86
Circle Area:           14897.07
Polygon Side Length:      80.95
Polygon Area:          11274.50
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
```

Example Execution #5:

```
1. Polygon inside of circle
2. Circle inside of polygon
Select your option -> 1
Enter number of polygon sides -> 12
Enter length of polygon apothem -> 9.93

-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
Circle Radius:            10.28
Circle Area:             332.02
Polygon Side Length:       5.32
Polygon Area:            317.05
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
```

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 `double` type.
    - Use the constant value `M_PI` where needed for the constant pi.
    - The number of sides of the regular polygon will always be an **integer** greater than 3.
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.
    - Any use of logical operators, relational operators, `bool` variables, or selection constructs is **prohibited** and would violate requirements of this assignment **resulting in no credit being awarded for your effort.**
    - **All code for this assignment will be placed inside of the `main` function.** User-defined functions will be a requirement beginning with lab #4.
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: `lab03.c`

Course Programming and Documentation Standards Reminders:

- Maximize your use of symbolic/defined constants and minimize your use of literal constants.
- 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.