CS 159 – Spring 2022 – Lab #6

Overview of Current Lab:

1. **Important** – submit your attendance at the start of every lab meeting!
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 introduced in lecture.
4. Begin your work on the programming assignment.
5. **New lab partners, ensure you have their contact information.** 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 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. Supporting documentation (structure charts, flowcharts) may be requested when seeking assistance from course staff.

- **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 #6 – 5 points possible: The lab quiz will be made available on Brightspace (Week #7 module) during the final 10 minutes of your lab today. The quiz will emphasize material from chapter 5 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 or in an unexpected location are subject to loss of points and a potential academic integrity inquiry.

Lab Programming Assignment #6 – 5 points possible: Selection is here! Writing and evaluating selection constructs requires planning to avoid unnecessarily long constructs and avoiding duplication of logic in multiple places in the program.

- **How might collaboration be useful on this particular programming assignment?** Investment in your algorithm now will save countless hours of frustration or an inefficient algorithm (poor design). Have your structure charts and flowcharts ready to share with course staff when requesting assistance.

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 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 5 and the course standards.

Statement	True or False
The use of any technique of repetition would violate requirements of this assignment and result in no credit being awarded for your effort.	**TRUE**
Section 5.1	
A piece of data is called logical if it conveys the idea of true or false.	
C programmers use other types, such as integers, to represent logical data.	
If a data value is zero it is considered false, but any non-zero value is considered true.	
The logical AND (`&&`) operator is true only when both operands are true.	
The logical OR (`||`) operator is true only when exactly one of its operands is true.	
The AND and OR operator share the same level of operator precedence.	
The short-circuit method of evaluating logical expressions will stop evaluating the current expression as soon as the result can be determined.	
The complement of the equal operator is the not equal operator.	
The complement of the greater than operator is the less than operator.	
When attempting to print the result of a logical expression that is true as an integer the result will always be 1.	
Section 5.2	
The logical expression of an `if...else` construct must be enclosed in parentheses.	
There is no semi-colon that follows the logical expression of an `if...else` construct.	
The statements found inside of an `if...else` may be any statement, including another `if...else` construct.	
The expressions `if(a != 0)` and `if(!a)` are complements.	
The expressions `if(a == 0)` and `if(a)` are complements.	
The dangling `else` logical error can be corrected by indenting the `if` and `else` the same number of spaces.	
The conditional expression has three operands and a two-token operator.	
Conditional expressions cannot be nested as the resulting code becomes too complex.	
`3 && -3 && 10 && -10`	1
`3 || -3 || 10 || -10`	1
`3 || 6 && 0`	1
`3 == 4 || 6 || 8`	0
`3 == 3 && -1 && 1`	0
`6 % 2 || 7 % 2`	0

Solve the following problems related to material found in Chapter 5 and the course standards.

Statement	True / False
The following two logical expressions are equivalent for all integer (`int`) values of `x`: `!(x < 10)` and `x >= 10`	
The following two logical expressions are equivalent for all non-negative integer (`int`) values of `x`: `x % 2` and `x % 2 != 0`	

The complement of
`x % 3 == 0 || x % 3 == 2`
is
`x % 3 != 0 && x % 3 != 2`
for all non-negative integer (`int`) values of `x`.	

The complement of
`x > 0 && x < 10 || y + 2 == 0`
is
`x <= 0 || x >= 10 && y + 2 != 0`	

The compiler will issue a warning when an assignment operator rather than the equality operator is used as the logical expression of an `if` condition.	
It is a course standard to make use of `{` and `}` with all `if-else` constructs.	
It is a course standard to indent all code within the body of a selection construct two additional spaces.	

Identify a value of integer (`int`) variables `x` and `y` that demonstrate the following expressions are NOT complements:

`x > 0 && x < 10 || y != 0`

`x <= 0 || x >= 10 && y == 0`

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 #6 - 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**. With the introduction of user-defined functions the role of Navigator should include tracking function names, return types, and parameters to help the driver as they enter code.
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,** can be tested using a variety of input to verify correctness, and **complies with the additional requirements of the assignment.**

Problem: Given the lengths of four potential sides to create a triangle, select and display the three sides that create the largest and smallest possible area of a triangle. Use Heron's formula to calculate the area of a triangle.

Checkpoints During Lab #6:		**TA Verification**
1 – Planning	Begin with a structure chart and then flowcharts for those functions that will make use of selection. Look to maximize the benefits of user-defined functions.	
2 – Getting Started in Vocareum	File `lb06.c` created, assignment header completed, `main` function inserted, variables declared and commented, relevant symbolic/defined constants created.	
3 – Function Declarations and Calls	Add the function declarations to the global declaration section of the program. Complete the `main` function with the necessary function calls.	
4 – Implementation of Input Functions	Demonstrate input operates as expected. Other functions called in `main` should be made inactive.	
5 – Implementation of Calculations	Demonstrate functions related to required calculations operate as expected. Output functions may be inactive and replaced with diagnostic print statements during this time. • Allow each partner to contribute to the implementation.	
6 – Implementation of Output Functions	Final output matches expected, additional test cases created to further demonstrate correctness of logic.	

Example Execution #1 (sides are displayed in order input):

```
Enter length of side #1 -> 20
Enter length of side #2 -> 17.5
Enter length of side #3 -> 24.5
Enter length of side #4 -> 21

Largest possible triangle area: 201.19
Side values used: 20.0, 24.5, 21.0

Smallest possible triangle area: 161.95
Side values used: 20.0, 17.5, 21.0
```

Example Execution #2:

```
Enter length of side #1 -> 20
Enter length of side #2 -> 15
Enter length of side #3 -> 25
Enter length of side #4 -> 20

Largest possible triangle area: 195.16
Side values used: 20.0, 25.0, 20.0

Smallest possible triangle area: 139.05
Side values used: 20.0, 15.0, 20.0
```

Example Execution #3 (use of 200 is not possible as the sum of two sides cannot be less than or equal to the third):

```
Enter length of side #1 -> 34
Enter length of side #2 -> 22
Enter length of side #3 -> 200
Enter length of side #4 -> 27

Largest possible triangle area: 296.66
Side values used: 34.0, 22.0, 27.0

Smallest possible triangle area: 296.66
Side values used: 34.0, 22.0, 27.0
```

Example Execution #4:

```
Enter length of side #1 -> 100
Enter length of side #2 -> 500
Enter length of side #3 -> 25
Enter length of side #4 -> 400

Error: No triangles can be created from lengths given!
```

Example Execution #5:

```
Enter length of side #1 -> 30
Enter length of side #2 -> 29
Enter length of side #3 -> 30
Enter length of side #4 -> 30

Largest possible triangle area: 389.71
Side values used: 30.0, 30.0, 30.0

Smallest possible triangle area: 380.81
Side values used: 30.0, 29.0, 30.0
```

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 input will be positive numeric data and will not exceed what can be stored in a variable of the `double` type.
3. For this assignment you will be **required** to implement the user-defined functions (from chapter 4). Failing to follow course standards as they relate to good user-defined function use will result in a **zero for this assignment.**
4. Revisit **course standards as it relates what makes for good use of user-defined functions, what is acceptable to retain in the** `main` **function, and when passing parameters by address is appropriate.**
    - In many cases user-defined function use should result in a `main` function that only declares variables and makes function calls.
5. Course standards **prohibit** the use of programming concepts not yet introduced in lecture. For this assignment you can consider all material in the first **FIVE** chapters of the book, notes, and lectures to be acceptable for use.
    - Course standards **prohibit** the use of programming concepts beyond the material found in the first FIVE chapters of the book, notes, and lectures.
6. A program **MUST** compile, be submitted through Vocareum as demonstrated during the lab #0 exercise, and successfully submitted prior to the posted due date to be considered for credit. The C-file you submit must be named exactly: `lb06.c`, no variation is permitted.

Course Programming and Documentation Standards Reminders:

- Code found inside the body of relevant selection and repetition constructs must be indented two additional spaces.
- Make use of `{` and `}` with all relevant selection and repetition constructs.
- See page 258 of your C programming text regarding the proper indentation for a `switch` construct.
- Use the course function header (`vi` shortcut `:hfx` while in command mode) for every user-defined function in your program.
    - List and comment **all parameters** to a function, one per line, in the course function header.
    - **All function declarations** will appear in the global declaration section of your program.
    - **The user-defined function definitions will appear in your program after the** `main` **function.**
- Indent all code found within the `main` and all user-defined functions **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 a function.
    - At no point during the semester should these two sections ever overlap.
- 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.
- Maximize your use of symbolic/defined constants and minimize your use of literal constants.