CS 159 – Spring 2022 – Lab #4

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. 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 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 #4 – 5 points possible: The lab quiz will be made available on Brightspace (Week #5 module) during the final 10 minutes of your lab today. The quiz will emphasize material from chapter 4 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 #4 – 5 points possible: This is the first programming assignment that requires the use of user-defined functions. New terminology, syntax, and concepts make the topic of user-defined functions a challenging one that requires involvement at all stages of solution development to fully understand.

- **How might collaboration be useful on this particular programming assignment?** In advance of any code writing the group should identify the benefits of using functions and how your solution to this particular problem takes advantage of these potential benefits.

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? • **Note:** New teams next week, every student should come prepared with the resources necessary to complete the lab!			

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

Statement	True or False
The coding portion of lab #4 requires the use of the specific user-defined functions described in the assignment.	**TRUE**
Lab quizzes are individual efforts. No resources may be referenced during the lab quiz. Only one browser tab may be open during the lab quiz and any open second tab, or second browser, will be considered an academic integrity violation.	**TRUE**
The function call is an executable statement.	
The function declaration requires the data types and identifiers for each parameter.	
The function call requires the data types and identifiers for each parameter.	
The first line of the function definition requires the data types and identifiers for each parameter.	
The function definition contains executable statements that perform the task of the function.	
The first line of the function definition terminates with a semicolon (`;`).	
The value of a local variable may be returned through a `return` statement of a user-defined function.	
Parameters are defined as local variables in the function header (the first line of the function definition) and should not be re-defined within the local declaration section of the function.	
Additional local variables can be defined in the local declaration section of a function.	
A local variable cannot be referenced through its identifier outside of the function in which it is defined.	
A variable declared in the local declaration section of a function can have the same identifier as one of the parameters of the same function.	
Individual tasks in a program must be factored into individual user-defined functions.	
One benefit of user-defined functions is the potential reduction or elimination of duplicate code.	
A function assignment header for every user-defined function must be inserted immediately above the definition of the function it is documenting.	
Parameters being received by a function will be commented to the right of where they are defined.	
The `return` statement cannot contain an expression.	
Data sent from the calling function to the function being called will be received in the same order in which it was passed.	
The individual task represented by a function should be testable apart from the rest of the program.	
The sequence in which the statements are executed in a program can be altered through the use of functions and function calls.	
A user-defined function may be called more than once in a program.	
The control of the program always returns from the called function to the `main` function.	
A function may return at most one value.	
The role of the `main` function is to coordinate the function calls and to establish the data needs for a program.	

What is the output generated by problem #26 (page 222-223 of your C programming text**)?**

Given the `main` function below, write the function declaration for each user-defined function called by `main`:

```
int main()
{
  float income;
  float rate;

  income = getIncome();
  rate = calcRate(income);

  displayResults(income, rate);

  return(0);
}
```

Given the `main` function below, write the function declaration for each user-defined function called by `main`:

```
int main()
{
  float annualRate;
  int term;
  float principal;
  float moPay;

  annualRate = getAnnualRate();
  term = getTerm();
  principal = getPrincipal();

  moPay = calcPayment(annualRate, principal, term);

  printAnalysis(term * 12, annualRate, principal, moPay);

  printAlternatives(term, annualRate, principal);

  return(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 #4 - Programming Assignment
Due: 30 minutes prior to the start of your next lab meeting.
5 Points Possible

Collaborative Roles for the Lab Session

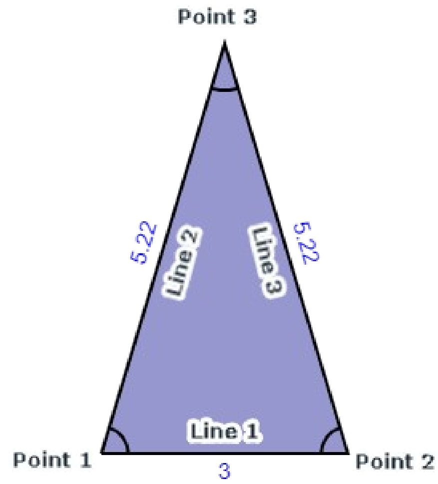
Collaborative Teaming. For this lab you will be working in your assigned teams. If you are unable to complete your assignment during the lab then it is expected that your team meet and collaborate outside of class to finish and submit the problem assigned. **Be sure to exchange contact information during 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 three distinct points that represent the corners of a triangle calculate the length of each side and interior angle. All coordinates are assumed to be floating-point (`double`) data and the three points will never be on the same line.

Checkpoints During Lab #4:		**TA Verification**
1 – Identify Tasks and Formulas	What will be the user-defined functions in your program? How do these functions take advantage of the benefits provided by function use? Correctly identify the formulas needed to solve this problem.	
2 – Getting Started in Vocareum	File `lb04.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.	
5 – Implementation of Calculations	Demonstrate functions related to required calculations operate as expected.	
6 – Implementation of Output Functions	Final output matches expected, additional test cases created to further demonstrate correctness of logic.	

Example Execution #1:




```
Enter coordinates for point #1 -> 0 0
Enter coordinates for point #2 -> 3 0
Enter coordinates for point #3 -> 1.5 5

-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
Line #1 Distance:   3.0 Angle:  33.4
Line #2 Distance:   5.2 Angle:  73.3
Line #3 Distance:   5.2 Angle:  73.3
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
```

Example Execution #2:

```
Enter coordinates for point #1 -> 0 0
Enter coordinates for point #2 -> 4 0
Enter coordinates for point #3 -> 3 3

-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
Line #1 Distance:   4.0 Angle:  63.4
Line #2 Distance:   3.2 Angle:  45.0
Line #3 Distance:   4.2 Angle:  71.6
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
```

Example Execution #3:

```
Enter coordinates for point #1 -> -1 0
Enter coordinates for point #2 -> -1 3
Enter coordinates for point #3 -> 3 0

-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
Line #1 Distance:   3.0 Angle:  36.9
Line #2 Distance:   5.0 Angle:  90.0
Line #3 Distance:   4.0 Angle:  53.1
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
```

Example Execution #4:

```
Enter coordinates for point #1 -> -5 1
Enter coordinates for point #2 -> -4 -2
Enter coordinates for point #3 -> -1 -1

-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
Line #1 Distance:   3.2 Angle:  45.0
Line #2 Distance:   3.2 Angle:  45.0
Line #3 Distance:   4.5 Angle:  90.0
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
```

Example Execution #5:

```
Enter coordinates for point #1 -> 1 1
Enter coordinates for point #2 -> 2 11
Enter coordinates for point #3 -> 3 0

-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
Line #1 Distance:  10.0 Angle:  58.2
Line #2 Distance:  11.0 Angle: 110.9
Line #3 Distance:   2.2 Angle:  10.9
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
```

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.
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 **four** chapters of the book, notes, and lectures to be acceptable for use.
6. 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: `lb04.c`

Course Programming and Documentation Standards Reminders:

- 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` 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.