CS 159 – Spring 2022 – Lab #10

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 (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 #10 – 5 points possible: The lab quiz will be made available on Brightspace (Week #12 module) during the final 10 minutes of your lab today. The quiz will emphasize material from chapter 8 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 #10 – 5 points possible

- **How might collaboration be useful on this particular programming assignment?** How will you organize the data generated in this problem? Review together your understanding of array declarations and the use of arrays with user-defined functions.

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 8 and the course standards.

Statement	True or False
Section 8.2	
In a fixed-length array the size of the array is known when the program is written.	
Arrays must be declared and defined before they can be used.	
Array declarations will determine the type, name, and size of the array.	
For a value to be potentially used as an index it must be an integral value or an expression that evaluates to such.	
The name of an array is a reference to the address of where it begins inside the memory of the computer.	
The index value represents an offset from the beginning of the array to the element being referenced.	
Declaration and definition of an array will include a default initialization of all elements.	
If the number of values provided for initialization of an array is fewer than the size of the array then the remaining elements have no known value.	
The address operator is not necessary in a `scanf` to accept input for an individual array element when using the indexing technique.	
When accessing an array element the C language does not check whether the index is within the boundary of an array.	
Section 8.3	
Arrays can be passed in two ways; by individual elements or the whole array.	
Elements of an array, themselves individual values of a given data type, are passed by value from calling to called function.	
The called function cannot identify whether the value it receives comes from an array, an individual variable, or an expression that evaluates to the expected type.	
Individual elements of an array can be passed by address through the use of the address operator.	
The reason that the C language does not pass whole arrays by value is the extra stress it would put on the memory of the computer to make a copy of an array.	
The name of an array is a primary expression whose value is the address of the first element in the array.	
Indexed references to individual elements of an array are simply calculated addresses where the index value is added to the address represented by the name of the array.	
Passing the array name to a function allows changes in the called function to be available back in the calling function after it terminates.	
When passing a whole array to the function the total size of the array is necessary in the function call.	
When passing a whole array to the function the total size of the array is necessary in the definition of the called function.	
It is only the starting point of the array in memory that is represented by the name of an array and not the ending point.	

What are the values in the array after the code below has been executed?

```
int x[5] = {1, 3, 7};
int i;

for(i = 0; i < 4; i++)
{
  x[i + 1] += x[i];
}
```

```
int x[5];
int i;

for(i = 0; i < 4; i++)
{
  x[i] = 2 * i - 1;
}
```

Statement	True / False
It is a course standard to make use of a symbolic/defined constant to represent the size of a statically declared array.	
The conversion code to use for input or output of an array element depends on the data type of the array.	
Variables and loops are commonly used together to generate index values to access the elements of an array.	
Arrays in the C programming language use a one-based index.	
To pass the whole array to a function you need to use the name of the array followed by empty square braces `[]` in the function call statement.	
All elements of one array can be assigned to another through the use of the assignment operator and the name of each array (example: `x = y`).	
While the default technique of passing array elements is by value it is possible to pass elements by address using the `&` operator (and the `*` operator in the function being called).	
If more than one element of an array are passed to a function in a single function call then those elements are passed by address.	
Using the name of an array in the data list of a single `printf` function will result in the output of all elements of the array.	
All arrays sent to a given user-defined function must be of the same defined size.	

Create a function (and function calls) to defend your answer to the final statement above:

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 #10 - 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.

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 a positive integer to serve as the seed for the random number generator, generate 500 two-dimensional points (generate x then y for each point) and analyze the data set created to calculate (1) the maximum distance between two points, (2) the number of horizontal lines, and (3) the number of vertical lines. The range of both dimensions will be integers in the range from -100 to 100 inclusive.

Point #		**1**	**2**	**3**	**4**	**5**
		(3, 5)	(0, -1)	(-4, 5)	(0, 8)	(3, 5)
1	(3, 5)		6.7	**7.0**	4.2	0.0
2	(0, -1)			7.2	**9.0**	6.7
3	(-4, 5)				5.0	**7.0**
4	(0, 8)					4.2
5	(3, 5)					

The example data on the left includes five points for the purpose of demonstration. Distances are calculated for each pair of points. There are two horizontal lines (red text) and one vertical line (blue text). A horizontal or vertical line must have a positive distance (greater than zero). It is possible that the data set may contain duplicate points. The maximum distance is between point #2 and #4.

Example Execution #1:

```
Enter seed value -> 3000

Maximum Distance: 265.2
Horizontal Lines: 609
Vertical Lines: 591
```

Example Execution #2:

```
Enter seed value -> 4000

Maximum Distance: 272.2
Horizontal Lines: 594
Vertical Lines: 624
```

Example Execution #3:

```
Enter seed value -> 0

Error! Positive seed values only!!

Enter seed value -> 5000

Maximum Distance: 266.2
Horizontal Lines: 636
Vertical Lines: 623
```

First ten points generated for each of the example executions:

Seed 3000	(-61, -48), (-30, 89), (33, 69), (-13, 68), (62, 89), (-89, 62), (-1, -39), (79, -44), (-8, 32), (86, -43)
Seed 4000	(-70, 44), (14, 57), (-2, -62), (60, -93), (21, -94), (-35, 15), (-35, 59), (-43, -50), (67, 90), (-17, -72)
Seed 5000	(10, -87), (-1, -72), (-88, 19), (-38, 35), (-53, -1), (-30, 18), (-28, 53), (-5, -14), (-19, 33), (100, 62)

All course programming and documentation standards are in effect for this and each assignment this semester.

Checkpoints During Lab #10:		**TA Verification**
1 – Planning	Begin with a visual representation of your data, a structure chart, and flowcharts for those functions that will make use of repetition.	
2 – Getting Started in Vocareum	File `lb10.c` created, assignment header completed, `main` function inserted, variables declared and commented, relevant symbolic/defined constants created.	
3 – Implementation of Functions	Use diagnostic print statements demonstrate functions related to required tasks operate as expected. Rotate between members operating the keyboard in between each successfully coded and tested function.	

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.
    - Input validation expectations are demonstrated in the third example execution.
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 previously presented material in the first **EIGHT** chapters of the book, notes, and lectures to be acceptable for use. See course standards below for array declaration expectations.
    - **The use of** any dynamic array structures (chapters 9 and 10) would violate this requirement and result in **no credit being awarded for your effort.**
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: `lb10.c`, no variation is permitted.

Course Programming and Documentation Standards Reminders:

- It is common to make use of a symbolic/defined constant when the size of the array is known prior to the start of a program.
- The course standards expect all arrays to be of a fixed size. Variable-size arrays, even those demonstrated in chapter 8 of the text, would violate course standards.
- 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.

Course Programming and Documentation Standards Reminders (continued):

- 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.

First ten points generated for each of the example executions:

Seed 3000	(-61, -48), (-30, 89), (33, 69), (-13, 68), (62, 89), (-89, 62), (-1, -39), (79, -44), (-8, 32), (86, -43)
Seed 4000	(-70, 44), (14, 57), (-2, -62), (60, -93), (21, -94), (-35, 15), (-35, 59), (-43, -50), (67, 90), (-17, -72)
Seed 5000	(10, -87), (-1, -72), (-88, 19), (-38, 35), (-53, -1), (-30, 18), (-28, 53), (-5, -14), (-19, 33), (100, 62)