CS 159 – Spring 2022 – Lab #11

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 #11 – 5 points possible: The lab quiz will be made available on Brightspace (Week #13 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 #11 – 5 points possible

- **How might collaboration be useful on this particular programming assignment?** How will you organize the data generated in this problem? What is the basis for your selection of sorting logic in this problem? All three algorithms are coded in the book but need to comply with course standards, what are those necessary changes?

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.

Statement	**True or False**
To conserve memory the sorting of data is completed within the array rather than creating a second array.	
Both the sorted and unsorted lists exist within the same array as the process of sorting data continues.	
The goal of each sorting algorithm is to move data from the unsorted list to the sorted list of an array.	
A sort pass is the process of moving an element from the unsorted list to the sorted list.	
Several sort passes will be necessary to sort most data sets.	
Section 8.5	
If we have a list of `N` elements, we need `N - 1` passes to complete rearrange the data (in sorted order).	
Exchanging the `'<'` for a `'>'` in the code on line 21 of page 495 of the C programming text will sort the data in the array from largest (index 0) to smallest (index of `last` minus one).	
From an efficiency point of view, it makes no difference whether the data is ultimately sorted largest to smallest or smallest to largest in an array.	

Additional content related to lab quiz #11:

1. Review the code provided on page 170 of your course notes packet.
2. Review the problem and solution provided from page 173 of your course notes packet.
3. Know how to determine if an integer is a semi-square.

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 #11 - 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 and the maximum value to generate, **create a data set of 100 integers** ranging from 1 to the given maximum value (inclusive of both end points) and display the perfect squares (in sorted order) followed by the semi-squares (in sorted order). A number identified as a perfect square will not be a candidate for display as a semi-square.

Example Execution #1:

```
Enter seed value -> 1000
Enter maximum range value -> 10000

Perfect squares: 4624
Semi-squares: 1922 4056 4205
```

Example Execution #2:

```
Enter seed value -> 1000
Enter maximum range value -> 100

Perfect squares: 1 1 1 1 4 4 16 36 49 100 100
Semi-squares: 18 32 48 48 48 72 75 98
```

Example Execution #3:

```
Enter seed value -> 2000
Enter maximum range value -> 25000

Perfect squares: 5929 11449 12769
Semi-squares: 1445 1690 10625
```

Example Execution #4:

```
Enter seed value -> 2000
Enter maximum range value -> 40000

Semi-squares: 23275 25088 32258
```

Example Execution #5:

```
Enter seed value -> 3000
Enter maximum range value -> 17

Perfect squares: 1 1 1 1 1 1 1 4 4 4 4 4 4 4 4 9 9 9 9 9 9 9 9 16 16 16 16 16
```

Example Execution #6:

```
Enter seed value -> 118968
Enter maximum range value -> 50000

Semi-squares: 147 1176 1805 11774 14297 21294 39375 40817 41971 44402
```

Example Execution #7:

```
Enter seed value -> -1

Error! Positive seed values only!!

Enter seed value -> 5000
Enter maximum range value -> 0

Error! Positive range values only!!

Enter maximum range value -> 200

Perfect squares: 4 16
Semi-squares: 32 72 98 147 180 200
```

Example Execution #8:

```
Enter seed value -> 1045
Enter maximum range value -> 15000

Result: No perfect or semi-squares found.
```

Example Execution #9:

```
Enter seed value -> 2000
Enter maximum range value -> 100000

Perfect squares: 12769
Semi-squares: 32258
```

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

Checkpoints During Lab #11:		**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 `lb11.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 seventh 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.

Additional Requirements (continued):

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.
    - Logic for each of the sorting algorithms introduced in the course can be found in the official text and notes packet. This code may be used in your assignments as long as you cite your source in the relevant function header and update all syntax to be in line with course standards.
    - **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: `lb11.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.

- 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.