CS 159 – Spring 2022 – Lab #7

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 #7 – 5 points possible: The lab quiz will be made available on Brightspace (Week #8 module) during the final 10 minutes of your lab today. The quiz will emphasize material from chapter 5, the generation of random numbers, 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 #7 – 5 points possible: More selection! Review the options available for each selection task and be sure you are making a good choice from among the available constructs.

- **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 4, Chapter 5, and the course standards.

Statement	True or False
Section 4.5 - Random Numbers	
The first random number generated is based on a seed, either the default provided by the system or one specified by the programmer.	
The `srand` function creates the starting seed for a number series based on the value it receives.	
The `srand` function must be called only once for each random number series.	
The random number function returns an integer between 0 and `RAND_MAX`, which is defined in the `stdlib.h` library as the largest number that `rand` can generate.	
While each call to the `rand` function returns the next integer in the series, the `srand` function is a `void` function.	
To generate a random integer in a range `x` to `y`, we must first scale the number and then, if `x` is greater than 0, shift the number within the range.	
The following expression is used to generate a positive integer value in the range from `x` to `y`, `rand() % (y - x) + x`	
Section 5.3	
The `switch` construct can only be used when the selection condition reduces to an integral expression.	
The control expression that follows the keyword `switch` may be a character expression.	
When the selection is based on a range of values, or the condition is not integral, we use the `else-if` for our multiway selection needs.	
The `case` label represents an integral type value that is a possible result of the control expression.	
Each `switch case` label is the keyword `case` followed by a constant expression inside of single quotes.	
Associated with a `case` label is zero or more executable statements.	
When the statements associated with one `case` have been executed the program flow continues with the statements for the next `case`.	
The course standards limit the use of the `break` statement to only `switch` statements.	
No two `switch case` labels can represent the same constant expression value.	
It is always a logical error to associate two `switch case` labels with a common set of actions.	
The maximum number of actions that can be associated with a `switch case` label is one.	
The `break` statement results in the control of the program exiting the `switch` statement.	
In an `else-if` the `if` condition is evaluated first and additional `else-if` conditions are evaluated until a condition is found to be true.	
Only the statements associated with the first true condition are executed in a multiway `else-if` construct.	
The `else` is executed only when all previously evaluated conditions are false.	
The `else` does not have a condition associated with it.	

Section 5.6	
Negative logic refers to any expression that begins with a NOT operator or that contains multiple NOT operators within.	
Complementing a condition is one way to potentially remove negative logic from an expression.	
The only way to complement a NOT operator is with another NOT operator.	
When writing a selection construct the most probable conditions should come before those that occur less frequently.	
It is possible to indicate on a structure chart when a user-defined is called from within a section construct.	

Solve the following problems on pages 292-293 of your C programming text: 30, 31, and 32.

Consider using this space to develop a structure chart for this problem.

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 #7 - Programming Assignment
Due: 30 minutes prior to the start of your next lab meeting (March 22 – March 25).
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 an integer to serve as a seed value for the random number generator, begin by generating two integers that will represent the first two cards drawn from a standard deck of 52 cards for the dealer playing a game of 21. Should the total of the cards held by the dealer be less than 16 then the dealer will draw an additional card. The dealer will stop, or hold, should the total be 16 or greater. The dealer wins when the total is 21 and always loses when the total exceeds 21. The maximum number of cards that the dealer can draw is five.

The cards numbered 2 – 10 are worth their given value and the face cards (11 – 13, Jack, Queen, King) are worth 10. Any aces (card numbered 1) can be either worth 1 or 11.

The suits of the deck are represented in the following order, the first 13 are clubs, the next 13 are the diamonds, followed by the 13 hearts and final the final 13 are the spades.

Details on Random Number Generation: The problem which follows requires the use of the random number generator as found in chapter 4 of your C programming text (see page 191). For it to be possible to replicate our examples below your program must first accept a "seed" integer value from the user. More information on how seeding works can found in example 4-13 of your C programming text.

Example Execution #1:

```
Enter the seed value -> 135

The first card is #: 47 (8 of Spades) Current score: 8
The second card is #: 33 (7 of Hearts) Current score: 15
The third card is #: 37 (Jack of Hearts) Current score: 25

Dealer busts.
```

Example Execution #1 Explained:

- The first two cards drawn by the dealer results in a total of 15. The dealer must draw a third card. However, this third card exceeds the limit of 21 and the dealer busts (and loses).

Checkpoints During Lab #7:		**TA Verification**
1 – Planning	Begin with a structure chart and then flowcharts for those functions that will make use of selection. Seek to maximize the benefits of user-defined functions.	
2 – Getting Started in Vocareum	File `lb07.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.	

Example Execution #2 (hold minimum of 16 generated by the initial two cards):

```
Enter the seed value -> 140

The first card is #: 45 (6 of Spades) Current score: 6
The second card is #: 52 (King of Spades) Current score: 16

Dealer holds.
```

Example Execution #3:

```
Enter the seed value -> 45513

The first card is #: 27 (Ace of Hearts) Current score: 11
The second card is #: 1 (Ace of Clubs) Current score: 12
The third card is #: 27 (Ace of Hearts) Current score: 13
The fourth card is #: 40 (Ace of Spades) Current score: 14
The fifth card is #: 28 (2 of Hearts) Current score: 16

Dealer holds.
```

Example Execution #3 Explained:

- Your program is not required to determine if a card has previously been drawn.
- The first ace is assigned a value of 11.

Example Execution #4:

```
Enter the seed value -> 95606

The first card is #: 14 (Ace of Diamonds) Current score: 11
The second card is #: 14 (Ace of Diamonds) Current score: 12
The third card is #: 1 (Ace of Clubs) Current score: 13
The fourth card is #: 1 (Ace of Clubs) Current score: 14
The fifth card is #: 12 (Queen of Clubs) Current score: 14

Dealer holds.
```

Example Execution #4 Explained:

- Your program is not required to determine if a card has previously been drawn.
- The first ace is assigned a value of 11, it is later converted to 1 to avoid the final card causing the dealer to bust.

Example Execution #5:

```
Enter the seed value -> 45

The first card is #: 47 (8 of Spades) Current score: 8
The second card is #: 16 (3 of Diamonds) Current score: 11
The third card is #: 28 (2 of Hearts) Current score: 13
The fourth card is #: 8 (8 of Clubs) Current score: 21

Dealer wins!
```

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 integer 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 **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: `lb07.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.