CS 159 – Spring 2022 – Lab #9

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 #9 – 5 points possible: The lab quiz will be made available on Brightspace (Week #11 module) during the final 10 minutes of your lab today. The quiz will emphasize material from chapter 6 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 #9 – 5 points possible

- **How might collaboration be useful on this particular programming assignment?** Identify the tasks that involve repetition. Paper and pencil are useful planning tools to assist in the development of the logic needed to classify a number as either a perfect square or semi-square number. Does each repetitive process call for pretest or post-test looping constructs? Is each process counter-controlled or event-controlled?

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 6 and the course standards.

Statement	True or False
The use of arrays, including character pointers to represent string data would violate requirements of this assignment and result in no credit being awarded for your effort.	**TRUE**
Chapter 6	
According to the course standards a `for` loop should only be used with counter-controlled processes.	T
According to the course standards if all three expressions are not needed in a `for` loop then you should instead make use of a `while` loop for your pretest looping needs.	T
A reasonable effort should be made to convert most `while` loops into `for` loops.	F
The number of times the update action is executed is equal to the number of times the loop control expression is evaluated in a `for` loop.	T
It is possible to update/change/alter the loop control variable of a `for` loop inside of its body.	T
You can make use of `x++`, `++x`, `x += 1`, and `x = x + 1` interchangeably as the update (third) expression of a `for` loop to increment the loop control variable.	T
The `gcc` compiler as used on Vocareum this semester will permit a variable to be declared and initialized in the first expression of a `for` loop. See examples on pages 318-319 of your C programming text.	F
This `for` loop will iterate 10 times: `for(i = 0; i < 10; i++)`	T
This `for` loop will iterate 5 times: `for(i = 12345; i != 0; i /= 10)`	T
This `for` loop will iterate 6 times: `for(i = 1; i <= 32; i * 2)`	F
The condition in a recursive function when which the recursive function calls stop is known as the base case.	T
Recursion should not be used with event-controlled processes as the result may be more function calls than the memory of the computer can accommodate.	T
Recursion is a repetitive process in which a function calls itself.	T
An iterative solution involves the use of a loop to solve a repetition problem.	T
Iterative solutions are always better than recursive ones.	F

What is the output generated by the code segments below?

```
int x;
int y;

for(x = 1; x <= 5; x++)
{
  for(y = 1; y <= 5; y++)
  {
    x == y ? printf("%d", x) : printf("X");
  }

  printf("\n");
}
```

```
int x;
int y;

for(x = 0; x < 5; x++)
{
  for(y = 1; y <= 5; y++)
  {
    if(y == 5 - x)
    {
      printf("%d", y);
    }
    else
    {
      printf("X");
    }
  }

  printf("\n");
}
```

```
 1 #include<stdio.h>
 2
 3 int calcSum(int);
 4
 5 int main()
 6 {
 7   int x = 35564;
 8   int sum;
 9
10   sum = calcSum(x);
11
12   return(0);
13 }
14
15 int calcSum(int x)
16 {
17   int sum = 0;
18
19   printf("Received value of x: %d\n", x);
20
21   if(x > 0)
22   {
23     sum = x % 10;
24     sum += calcSum(x / 10);
25   }
26
27   printf("Returned values of sum: %d\n", sum);
28
29   return(sum);
30 }
```

What is the output of the recursive program above?

How many total times if the `calcSum` function called in the program above?

What is the condition of the base case of the recursive function in the program 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 #9 - 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 newly 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 value as a starting point determine the next integer, starting with the value input, that meets one of following:

1. Is a perfect square such that the number entered (`b`) is equal to the square of another integer (`a`).

OR

2. Is a semi-square such that the numbered entered (`b`) is equal to the product of the square of another integer (`a`) and a third integer (`c`) where `c` is greater than 1 and less than `a`.

Example Execution #1:

```
Enter starting value -> 27

Identified value: 32
Status: Semi-square
```

Example Execution Explained:

Test values beginning with the number given and identify the first that meets the description of (1) or (2) provided above.

How was 32 identified as a semi-square?

The value `b` is 32, the value `a` is 4 (16 when squared), and the third integer `c` = 2 is greater than 1 and less than `a`.

Example Execution #2:

```
Enter starting value -> 99

Identified value: 100
Status: Perfect square
```

Example Execution #3:

```
Enter starting value -> 701

Identified value: 720
Status: Semi-square
```

Example Execution #4:

```
Enter starting value -> 37

Identified value: 48
Status: Semi-square
```

Example Execution #5:

```
Enter starting value -> 65

Identified value: 72
Status: Semi-square
```

All course programming and documentation standards are in effect for this and each assignment this semester. Please review this document!

Example Execution #6:

```
Enter starting value -> 15610

Identified value: 15625
Status: Perfect square
```

Example Execution #7:

```
Enter starting value -> 3481

Identified value: 3481
Status: Perfect square
```

Example Execution #8 (input validation demonstrated):

```
Enter starting value -> 0

Error! Positive values only!

Enter starting value -> -1

Error! Positive values only!

Enter starting value -> 3528

Identified value: 3528
Status: Semi-square
```

Checkpoints During Lab #9:		**TA Verification**
0 – Establish Team Expectations	Two big events remain (second midterm, final exam) that will go a long way to determining success in the course. How will your collaboration on lab tasks facilitate that success? Set specific expectations now!	
1 – Planning	Begin with a structure chart and then flowcharts for those functions that will make use of repetition. Seek to maximize the benefits of user-defined functions.	
2 – Getting Started in Vocareum	File `lb09.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 final example execution.
    - All input, output, and values generated in between will be of the `int` data 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.

Additional Requirements (continued):

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 **SIX** 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 SIX chapters of the book, notes, and lectures.
    - The use of arrays, including character pointers to represent string data would violate requirements of this assignment 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: `lb09.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.