CS 158/159 Lab 08

In this lab session you will complete the following tasks in the specified order. **Do not begin this lab until your lab sessions starts**. Individuals failing to remain on task during the lab will be assigned a zero for the lab, lab quiz, and will be considered absent for the lab. Web browsing, personal e-mail, social networking, and/or text messaging is not permitted during lab.

1. Record Attendance

Log on to your UNIX account on guru on a physical PC in the lab room (not on a wireless device) and enter the command **`attend`** to officially record your lab attendance. A confirmation e-mail will then be sent to you to verify that your attendance was correctly recorded. If you are late, you should still run the `attend` command and continue to participate in the remaining portion of the lab, but you will not be eligible to receive credit for the lab programming assignment. If you are present in the lab, but fail to run the `attend` command as expected, you will be marked as absent and likewise will not be eligible to receive credit for the lab programming assignment.

2. Review Material

The first task is to improve your understanding of the material from the book and the course programming standards (available in the course packet and in Blackboard) by working through a series of problems with the assigned lab partners of your group. You must show your lab instructor that you have successfully completed these tasks BEFORE you leave lab today. You will not receive points for the programming assignment, unless this task has been completed to the satisfaction of your lab instructor.

Solve the following problems related to material from Chapter 6:

Statement	**True / False**
An iteration is one evaluation of the loop control expression.	
The condition that determines whether iteration is to continue in a loop is known as the loop control expression.	
The loop control variable is commonly a part of the loop control expression and the recipient of the loop update action.	
The update to the loop control variable is responsible for terminating the loop because it will eventually result in the loop control expression in being false.	
In a post-test loop the minimum number of times that the statements found inside of the loop are executed is one.	
The initialization of the loop control variable must take place outside (before) a pretest loop.	
Course standards prohibit the use of `break` to terminate any repetition construct.	
A limited amount of control structures are permissible in the `main` function.	
An infinite loop is a logical error.	

The only way to stop a program caught in an infinite loop is to shut down your terminal software.	
A nested loop is a repetitive process contained inside of another repetitive process.	
One approach to potentially make solving problems that require nested loops easier is to separate each repetitive process into its own function.	

Use the tables below to trace the execution of the following loops:

```
int x = 3415263;
int ct = 0;

while(x > 0)
{
 if(x % 2 == 0)
  ct++;

 x = x / 10;
}
```

Iteration #	Value of X	Value of CT
1		
2		
3		
4		
5		
6		
7		
8		

```
int x = 3415263;
int ct = 0;

do
{
 x = x / 10;

 if(x % 2 == 0)
  ct++;

} while(x > 0);
```

Iteration #	Value of X	Value of CT
1		
2		
3		
4		
5		
6		
7		
8		

3. Programming Assignment

The second task is to develop a program as a group which solves the given problem. This assignment is worth 15 points and will be **due 30 minutes prior to the start of your next lab session**. All assignment deadlines are firm and the ability to submit your assignment will be disabled after your deadline elapses. No late work will be accepted!

As you develop the program, you should rotate through the following roles approximately every 30 minutes. Do not allow the same person be the driver the entire time. It is not acceptable to designate a single individual to complete the assignment. Every individual group member should have a full understanding of all work submitted. Assignments are an opportunity to develop and demonstrate your understanding of course material.

Role	Description
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 navigator's role is to look over the shoulder of the driver for syntactical errors, logical errors, and concerns related to course standards. The most common mistakes include failing to pair quotes, misplaced semi-colons, and improper placement of parentheses.
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 and can be tested using a variety of input to verify correctness. The manager is responsible for communicating to the teaching assistant who will be making the group's final lab submission.

This programming assignment does not have a single solution, and each group should collaborate together to develop their own unique solution. Submissions may be processed with comparison software and results will be used to detect unacceptable collaboration between groups. The development of your algorithm and the resulting code should only be discussed among your group members and course staff.

Your program must adhere to the course programming standards (available in the course packet and in Blackboard). Please review this document before submitting your assignment, because failure to adhere to it will result in a reduction in points. You program must include the designated program header (~cs15x/student/hdrProg) at the top of the program (which can be inserted in `vi` using the `hp` shortcut while in command mode). The header must include an appropriate description of your program and must contain the official Purdue career account e-mail addresses of each **contributing** group member. Do not include the e-mail address of anyone who did not actively participate in the program development. Failing to participate in the process to the satisfaction of all partners will result in a zero. Also note that course standards prohibit the use of advanced programming concepts not yet introduced in the course, unless otherwise specified.

Each of the example executions provided below represents a single execution of the program. Your program must accept input and produce output **exactly** as demonstrated in the example executions. Your program will be tested with the data seen in the examples below and an unknown number of additional tests making use of reasonable data. Do not include any example outputs with your submission.

A single program file (with the `.c` extension) must be submitted electronically via the `guru` server. An example submission was conducted during the first week in lab00. If you have a concern regarding how to submit work, please visit course staff prior to the assignment deadline.

Problem: Modify lab #7 so that the dealer will continue to draw additional cards while the total is less than 15. If the dealer does not bust (exceed 21 points) then the player will be given an option to draw additional cards until indicating that they wish to stop or should they bust. The old problem statement and details regarding how to draw cards from lab #7 can be found at the end of this problem.

Example Execution #1 (dealer draws one card for a total of 19, player draws three cards and busts)**:**

```
Enter seed -> 1239

Dealer's hand: 8 6
Your hand: 5 6

The dealer draws another card: 5
Dealer Note: hold.

Number of cards to draw (0 or 1) -> 1

You draw another card: 4
Number of cards to draw (0 or 1) -> 1

You draw another card: 2
Number of cards to draw (0 or 1) -> 1

You draw another card: 6
Dealer wins: player bust
```

Example Execution #2 (A score of 21 is referred to as a blackjack):

```
Enter seed -> 159

Dealer's hand: 7 5
Your hand: 5 3

The dealer draws another card: 9
Dealer Note: blackjack!

Number of cards to draw (0 or 1) -> 1

You draw another card: 5
Number of cards to draw (0 or 1) -> 0

Player note: hold.

The dealer wins: 21 to 13
```

Example Execution #3:

```
Enter seed -> 158

Dealer's hand: 6 2
Your hand: 2 3

The dealer draws another card: A
The dealer draws another card: 7
Dealer Note: hold.

Number of cards to draw (0 or 1) -> 1

You draw another card: 9
Number of cards to draw (0 or 1) -> 1

You draw another card: 3
Number of cards to draw (0 or 1) -> 0

Player note: hold.

You win: 17 to 16
```

Example Execution #4 (A score exceeding 21 is a bust)**:**

```
Enter seed -> 1244

Dealer's hand: 5 4
Your hand: 8 3

The dealer draws another card: 5
The dealer draws another card: 8

You win: dealer bust
```

Example Execution #5:

```
Enter seed -> 3125

Dealer's hand: A 2
Your hand: 7 J

The dealer draws another card: 2
The dealer draws another card: A
The dealer draws another card: 3
The dealer draws another card: 4
The dealer draws another card: 2
Dealer Note: hold.

Number of cards to draw (0 or 1) -> 0

Player note: hold.

You win: 17 to 15
```

Example Execution #6:

```
Enter seed -> 709291

Dealer's hand: K A
Your hand: A 4

The dealer draws another card: Q
Dealer Note: blackjack!

Number of cards to draw (0 or 1) -> 1

You draw another card: A
Number of cards to draw (0 or 1) -> 1

You draw another card: 8
Number of cards to draw (0 or 1) -> 1

You draw another card: A
Number of cards to draw (0 or 1) -> 1

You draw another card: A
Number of cards to draw (0 or 1) -> 1

You draw another card: 3
Number of cards to draw (0 or 1) -> 1

You draw another card: 2
Player note: blackjack!

It's a draw, score: 21
```

Example Execution #7:

```
Enter seed -> 84582

Dealer's hand: 7 2
Your hand: 7 3

The dealer draws another card: 8
Dealer Note: hold.

Number of cards to draw (0 or 1) -> 1

You draw another card: A
Number of cards to draw (0 or 1) -> 1

You draw another card: A
Number of cards to draw (0 or 1) -> 1

You draw another card: 9
Player note: blackjack!

You win: 21 to 17
```

Example Execution #8 (there is a chance that the random number generator will generate six consecutive multiples of thirteen)**:**

```
Enter seed -> 7754610

Dealer's hand: A A
Your hand: A A

The dealer draws another card: A
The dealer draws another card: A
The dealer draws another card: 7
The dealer draws another card: 2
The dealer draws another card: J

You win: dealer bust
```

Example Execution #9:

```
Enter seed -> 50715

Dealer's hand: J J
Your hand: 6 7

Dealer Note: hold.

Number of cards to draw (0 or 1) -> 1

You draw another card: 7
Number of cards to draw (0 or 1) -> 0

Player note: hold.

It's a draw, score: 20
```

Additional Notes:

- All input can be stored in an integer (int) variable.
- You do not need to worry about invalid options entered by the user.
- Course standards prohibit the use of programming concepts not yet introduced in lecture. For this assignment, you can only consider material in the first 6 chapters of the book, notes, and lectures. Use of advanced programming constructs beyond this material would result in a loss of points.

Problem from Lab #7: Boiler blackjack is a two player card game in which each player, one of which is called the dealer, receives two cards with the option to select a third card in an effort to reach, but not exceed, the score of 21. The dealer will receive their two cards first and then two cards are given to the other player. It will be the dealer who must decide first whether they would like a third card or to hold at the two original cards. The dealer will always accept a third card whenever the total of the two original cards is less than or equal to 16. After the dealer has completed their turn and if the score of the dealer remains less than 22, then the other player will determine whether to accept a third card via input.

4. Group Coordination

The next task is to collaborate as a group to determine who will submit your program assignment for grading. **Only one person per group will make submissions for the entire group**. Lab instructors are not required to grade submissions from multiple members of the same group to determine which submission you actually want graded. Also, the group member should not always be making the submission each week. Record the names and official Purdue career account e-mail addresses of all three lab partners here and put a checkmark in the Submitter column for the person responsible for making the submission. Your group should turn in this page of information to their lab instructor before you leave lab today.

Name	Purdue career account e-mail address	Submitter

If possible, it is a good idea to submit your work for grading prior to leaving lab today, even if it is not completely finished. This will allow each lab partner to **verify their contact information in the assignment header is correct** as each would then receive an e-mail verifying the submission. You may make multiple submissions before the deadline, but only the last attempt is retained and graded. Any previous submissions will be over-written and cannot be recovered. The submission script will reject the submission of any file that does not compile. A program must compile to be considered for partial credit.

If your group is unable to complete your assignment during the lab session, then it is expected that your group will meet outside of class to finish and submit the programming assignment. That is why you should exchange contact information during lab today! Before you leave, you should discuss when and where the group will meet next and when you plan for final submission to be made. If a group member you have entrusted to do the submission cannot be contacted and he/she is the only one who has a copy of the program, then the rest of the group would have to essentially start over in order to complete the programming assignment. Thus, it is a good idea for each person to have a copy of the program.

Consider the members of your group to be resources to assist you as you learn the material in this course. As a group, you may want to arrange a time to visit the TA office hours together or attend a particular Supplemental Instruction session together.

5. Lab Quiz

The final task will be to take the lab quiz which be made available in Blackboard during the last part of your lab today. **Lab quizzes are individual efforts and you may not use any resources while completing the quiz.** The quiz is worth 5 points and you must be present in lab session in order to credit for the quiz. All problems on lab quizzes will be multiple-choice or true-false. The quiz will emphasize material from book and the appropriate course programming standards. You will be given 10 minutes to complete the quiz and questions will be 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. Any quiz that is taken when it is not being proctored by the lab instructor will receive zero points.