**EECE 1080/ CS 1 - Laboratory 4**

***Game of Chance***

***Topics covered: Input Validation, Decision Statements, Loops, and Functions***

## Due:

Sunday Lune 9, 2019 by 11:59PM

## Objective:

Gain experience with functions and modular program to promote code reuse.

**Collaboration:**

As with most laboratory assignments in this course this laboratory assignment is to performed by an individual student. You can help each other learn by reviewing assignment materials, describing to each other how you are approaching the problem, and helping each other with syntax errors. You can get help from tutors, teaching assistants, and instructors. Having similar code for some assignments is expected but most assignments have multiple different solutions. Your code is expected to be different.

Please document any help you recieve from other students, teaching assistants, or instructors. Just add the names to the top of your source file.

Having someone else code for you, sharing code with other students, or copy-pasting code from the internet or previous terms, **is cheating**. A helper should "teach you to fish, not feed you the fish". Laboratory assignments prepare you for exams so be smart.

**Highlights**:

* Please review chapter on functions
* Please read the specification below carefully.
* Review **Grading Rubric** at the end of this document.
* Review **Sample Output** at the end of this document.
* Please ask appropriate questions when necessary.
* *To recieve full credit you will need to get Tasks 1 through 3 verified during the laborarory session. This will count towards your laboratory session attendance grade.*
* *Complete and have your program verified by the end of your scheduled laboratory session to recieve 10-bonus points.*
* Your program should use **at least 5 functions.**
* Your program source code should be formatted properly
* Your program should compile without warnings.
* Please submit the final result to blackboard for grading. The filename should be game.cpp.

## Specification:

Your program will run a simulation of a game of chance. The code developed in this lab should all be a part of a single program. Each task’s code will build upon previous tasks to create the final program at the end of task 5. **So you only have one program to submit at the end of this lab**.

Consider the following simple dice game

**Game Rules:**

You pick a number from 1 to 6, and call this number your point. You then roll a die 3 times. If there is any match with your point, then you win $1; otherwise you pay the house $1.

## Program Development:

You will be creating ***at least 5 functions*** to create this game program. Two of these functions will be **rollDie()** and **playOneGame()** as outlined in Tasks one through three below. The other three are your choice.

Some basic suggestions for functions might include the following:

* functions to perform input.
* functions to display statistics.
* a function to handle the complete logic of the program (move all or nearly all logic out of main())

The functions will need to make sense in the context of the program.

**Notes**:

* Avoid using non ***constant*** global variables.
* Source code should be properly formatted and consistently styled.
* You should comment the code appriopratiely.
* All program input should be verified for proper range of values. Have other users test your input.
* All program output should be probably formatted and readible. Have other students/users look at your output.

### Task 1:

Design a program to simulate one play of this game. You will need a simple function ***rollDie()*** that returns a random integer in range 1-6. You should prompt for the point and roll the dice up to three times to see if you win/lose.

|  |
| --- |
| Remember to utilize random numbers you need to do three things.   1. #include <cstdlib> 2. #include <ctime> 3. srand(time(NULL)); // Seed the random number, do this once, in main 4. rand() // Call rand to get a random number between 0 and INT\_MAX |

The rand() function does not take any parameters and returns an integer between 0 and 32K. You will need to use the modulus operator to scale it down to numbers between 1 and 6.

### Task 2:

If your game works correctly you should then design a loop control structure that keeps repeating the game, and tracks the number of wins and losses. The loop should continue until the user enters in an appropriate sentinel value, such as the character 'x' for exit. Verify that the program runs properly, and as expected, before continuing.

### Task 3:

Modify the code so that you encapsulate the logic of the game into a single function. The function should have a prototype:   
  
 ***bool playOneGame()***

The function should return true if the game is won by the user, and returns false if the game is lost. Note that you do not need to change the basic logic of the original program to design your function. Test your function with a driver program that prints the results of the function call.

### Task 4:

Modify the program so that the user is able to maintain a bank account (which is initialized with $100.00) and wager on each play. The program should prompt the user to enter a wager. This wager replaces the simple case of winning/losing $1 from before. Do an error check (using a while loop), so that the wager entered is no more than the current bank balance. If the wager is valid, then inform the user that they will be playing a single game. If the wager is invalid, print an error message and repeat the process.

### Task 5:

A play of a single game is made by calling the function you created in Task 2. A win (return value true by function playOneGame()) will increase the bank balance by the wager, and likewise a lose (return value of false by function ***playOneGame()***) will decrease the balance by same. After the game is played, print the current balance in an appropriate format. If the balance is greater than zero, prompt the user to play again. Continue another round of the game until the user responds that the user finishes playing (by wagering 0), or if the user’s balance is zero.

After the game is over you should print out the number of games won or lost, and the percentage of games won. You may want to create a function to output these statistics.

### Task 6:

Submit final solution to blackboard.

***Good Luck!***

## Example Program Run

Remember - randomization is in effect here, so subsequent runs will not have the same exact result.

|  |
| --- |
| Account balance $100  Enter wager (0 to exit): 50  Enter your point value (1 - 6): 1  Roll #1 is 1  \*\* You win! \*\*  Account balance $150  Enter wager (0 to exit): 50  Enter your point value (1 - 6): 1  Roll #1 is 5  Roll #2 is 4  Roll #3 is 5  \*\* You Lose \*\*  Account balance $100  Enter wager (0 to exit): 50  Enter your point value (1 - 6): 1  Roll #1 is 4  Roll #2 is 2  Roll #3 is 6  \*\* You Lose \*\*  Account balance $50  Enter wager (0 to exit): 50  Enter your point value (1 - 6): 1  Roll #1 is 5  Roll #2 is 1  \*\* You win! \*\*  Account balance $100  Enter wager (0 to exit): 100  Enter your point value (1 - 6): 5  Roll #1 is 4  Roll #2 is 2  Roll #3 is 6  \*\* You Lose \*\*  Game over, you are out of money!  Your final account balance is $0  You won 1 time out of 4 for a winning percentage of 25% |

Another sample run, where the user wagers 0 to exit.

|  |
| --- |
| Account balance $100  Enter wager (0 to exit): 75  Enter your point value (1 - 6): 4  Roll #1 is 2  Roll #2 is 4  \*\* You win! \*\*  Account balance $175  Enter wager (0 to exit): 0  Your final account balance is $175  You won 1 time out of 1 for a winning percentage of 100% |

And, another sample run, with input errors.

|  |
| --- |
| Account balance $100  Enter wager (0 to exit): -4  Error: You must wager between $1 and $100 (type 0 to exit): 101  Error: You must wager between $1 and $100 (type 0 to exit): 45  Enter your point value (1 - 6): -1  Error: Enter your point value, must be between 1 and 6: 1723  Error: Enter your point value, must be between 1 and 6: 234  Error: Enter your point value, must be between 1 and 6: 7  Error: Enter your point value, must be between 1 and 6: 6  Roll #1 is 3  Roll #2 is 1  Roll #3 is 3  \*\* You Lose \*\*  Account balance $55  Enter wager (0 to exit): 0  Your final account balance is $55  You won 0 time out of 1 for a winning percentage of 0% |

**You must submit your code to blackboard with the proper name (game.cpp).**

**Grading Rubric:**

* Proper source code styling: 10-Percent
* Proper commments: 5-Percent
* No compiler warnings: 5-Percent
* **rollDie() function** works properly: 5-Percent
* **playOneGame() functions** works properly: 20-Percent
* The game plays appropriately: 25-Percent
* The input is verified: 10-Percent
* The output is formated properly: 10-Percent
* Your program uses at least 5-functions 10-Percent

***Programs that do NOT*** c***ompile will only recieve “style” points.***

**Grading Policy for this Assignment:**

Your lab “milestone” attendance grade will include the following requirements:

1. Successfully implement the *rolldie()* function as described in Task 1.
2. Successfully implement the *playonegame()* function as described in Task 3.

For this assignment in particular, there will be bonus points available to students who achieve the following requirements:

1. Successfully implement Tasks 1-5 in the lab description **within your lab period**.
2. Have a TA check your finished code to ensure that it fully works **within your lab period**.
3. Upload the finished code to blackboard **prior to leaving your lab period**.

Students who successfully achieve all three objectives listed above will be rewarded **10 extra points** towards their lab assignment grade for this assignment. Students who finish the lab assignment and submit after their lab period will only be able to receive a maximum grade of 100% on the assignment, while students who successfully complete all objectives described above in lab will be rewarded with a 110%.