The University of Alabama in Huntsville
Electrical and Computer Engineering
Project 7 (40 points, perfect match +10% bonus)

Submit Your Solution Using Canvas by 10:00 PM, Friday October 18, 2019 (late submissions will be accepted on 10/18/19 from 10pm to 11:59pm)

<Project 7 Description>

You will write a program using functions that performs the following tasks. These tasks are explained in more detail later in this document

(1) For a user specified number of iterations, determine an approximation for pi using the Monte Carlo Method. Set the output precision to 6 decimal places for this part of the program.
(2) Determine the outcome of a **fair*** coin being flipped. For a user specified number of coin flips, determine the percent of the time that heads comes up and the percent of the time tails comes up. Set the output precision to 4 decimal places for this part.
(3) Determine the outcome of a **fair*** 5-sided die being rolled. For a user specified number of rolls, determine the percent of the time that each side (sides 1,2,3,4 and 5) come up. Set the output precision to 4 decimal places for this part.

***fair** – means that each possible outcome is equally likely – there is no bias for one outcome over the others. For the coin it means that heads has a 50% chance of coming up and so does tails.

In order to perform the above tasks, a random number generator is to be used to determine values for testing. Information regarding the random number generator occurs later in this document.

To perform the above tasks, your program should have the following order (pseudo algorithm/functional decomposition setup) as illustrated in the sample solution:

Prompt the user for a seed value for the random number generator
Write out a menu with the choices as shown in the sample solution and obtain the selection value
While the choice is not to exit, perform the following loop
    For choice 1, generate random numbers and process to determine the approximation for pi
    For choice 2, generate random numbers and process to determine the fair coin probabilities
    For choice 3, generate random numbers and process to determine the fair die probabilities
    Write out a menu for the next choice and obtain the selection value

<Project 7 Assumptions>

The seed value entered for the random number generator is a valid positive integer value and the number of iterations entered is a valid positive integer. **Largest integer value allowed is 2147483647**

<Project 7 Sample Solution>

You may run the sample solution **Project_07_solution** by typing the following at a command prompt in a terminal window**:**

Sample Solution – for determining vertical spacing differences

/home/work/cpe211/Executables/Project_07/Project_07_solution

Comparison Script – for determining line differences

/home/work/cpe211data/Project_07/CompareSolution.bash Project_07.cpp

To compare your results to the sample solution, both programs must be run on the same computer (The random numbers generated on one system will be different from those on a different system – even using the same start seed)

<Project 7 Directions>

On Project 7, you may only use concepts presented in Chapters 1 - 9 of your textbook

Using your favorite text editor, type your solution and save it as a file named **Project_07.cpp** within your **CPE211_FALL19/Project_07** directory. If there are syntax errors, correct them and compile again. Once your program successfully compiles, run it and verify that the output for your modified program matches the output from the solution executable – **Project_07_solution**.

Once you are satisfied with your solution, submit Project_07.cpp via Canvas.

NOTE: make sure that you do not change the order in which the information is entered. An automatic script is used to process all lab submissions, and if the order of the input information is modified, the script will not work properly with your program.

<Project 7 C++ Concepts Explained>

This program uses the random number generator in C++. To use the random number generator, the **header file cstdlib** is required. In order to compare results from the run of one program to another program, the same random numbers need to be generated by both programs. This is accomplished by supplying a seed value to start the random number generator at the same point. The code below performs the task of initializing the random number generator with a seed value:

```
// Setup the random number generator starting point by obtaining a seed
cout << "enter in the seed(integer > 0) for the random number generator: ";
cin >> seed;
cout << seed << endl; // echo print out the value entered
srand(seed); // use the seed entered to initialize the generator
```

Use srand(seed) one time only – at the beginning of the program. Random numbers are generated by calling the value returning function **rand()**. **rand()** returns an integer value between 0 and **RAND_MAX** (a constant with a value of 2147483647). In this program, all return values from **rand()** are converted to a floating-point number between 0 and 1 by dividing the returned value by **RAND_MAX**. An example of the type of line of code you will need to use is the following:

```
x_coord = double(rand())/double(RAND_MAX); // value is 0 to 1
```

Note the type casting to double on all of the integer values.

<Project 7 Function Requirements>

For this project, at least five functions will be written. **Function Prototypes must be used and function definitions must go below main. Read the project Addendum**. **The five required functions are**:

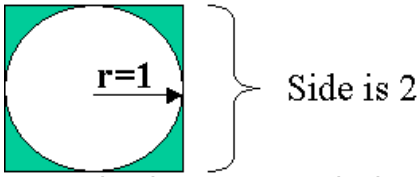
1) Print Menu – This function is to print out a menu of options. It is to do no other task. It should be a void function
2) Obtain Integer – This function obtains any integer value. It can be void or value returning. It handles all of the error correction and messages if any characters other than digits are entered. Use a switch statement or if-then-else–if statement in main to handle invalid integer values.
3) Calculate PI – this function calculates an approximation to PI using the monte-carlo method. This is a void function and it does not have any parameters
4) Flipping a Coin – This function determines the odds of heads or tails coming up when flipping a fair coin. This is a void function without parameters
5) Toss a Die – This void function without parameters, determines the odds of a 1, 2, 3, 4 or 5 coming up when a five sided fair die is tossed.

<Project 7 C++ Math Explained>

Monte Carlo Method: The Monte Carlo method is a way to use random numbers to approximate a value. In this program it is used to determine a value for pi. The background on the method is now discussed. Consider the figure below showing a circle of radius 1 inscribed inside a box of side length 2. The area of the

square is 2*2 or 4 units. The area of the circle is PI*radius*radius which is PI since the radius is 1. Therefore the area of the circle divided by the area of the box is PI/4. In the Monte Carlo method, two random numbers are used to represent an X-coordinate value (from 0 to 1) and a Y-coordinate value (from 0 to 1). The position of the X-Y coordinate pair relative to the circle is determined (is the pair inside the circle or outside the circle). Two counters are used. One counter for counting the number of co-ordinate pairs that fall inside ($\sqrt{X^2+Y^2}$ <=1.0) the circle, and one for the number of pairs that fall outside the circle Then the ratio of the number of co-ordinate pairs inside the circle to the total number of co-ordinate pairs tested approaches the ratio of PI/4. The more iterations performed, the closer the approximation becomes. For each loop iteration, an X and a Y co-ordinate pair is generated. The location of the X-Y pair relative to the circle is then determined. Once all loop iterations have completed, the approximate value for PI is calculated using the formula PI ~ 4*(number of co-ordinate pairs inside the circle)/(total number of iterations). Where ~ means *is approximately equal to*.

For the Monte Carlo method, 2 random numbers are generated for each loop iteration – one for the x value and one for the y value.



Coin Flip: For simulating the flipping of a coin, a random number between 0 and 1 is generated to represent each flip of the coin. If that value is **less than or equal to 0.5, then assume that the coin landed on tails** for **values greater than 0.5 the coin is assumed to have landed on heads**..

Rolling a Die: For simulating the rolling of a 5-sided die, a random number between 0 and 1 is generated to represent each roll of the die. If the random number is less than 0.2, then assume side #1 came up. If the random number is greater than or equal to 0.2 but less than 0.4, then assume side #2 came up. If the random number is greater than or equal to 0.4 but less than 0.6 then assume side #3 came up. . If the random number is greater than or equal to 0.6 but less than 0.8 then assume side #4 came up. For a random number greater than or equal to 0.8, assume that side number 5 came up.

For the coin flip and die roll parts of the program, one random number is generated for each loop iteration.

<Project 7 Message and Input Value Information>

The menu has 24 characters across the top and bottom lines
Invalid integer and invalid character messages have 47 characters across the top and bottom
For the PI output, there are 46 characters across the top and bottom of the output section
For the coin output, there are 47 characters across the top and bottom of the output section
For the die output, there are 45 characters across the top and bottom of the output section

➔ **The only input value that needs to be tested and verified is the menu choice.** ➔
➔ **Valid integer values will be supplied for the seed number and the number of iterations** ➔

<Project 7 Hints>

The function srand() is to be called one time only – at the beginning of the program. All random numbers are to be generated from this initial setting of the seed value.

Use double variables for the percent calculations for each of the methods.

Use a do-while loop in main to print the menu, obtain an integer, process the integer with a switch of if-then-else-if statement. Loop exits if the exit option is selected.

<Project 7 Restrictions>

Use of global variables will result in a grade of 0 on the project.
No arrays are allowed. Only concepts in Chapters 1-9 are allowed.