Lab Partner 1 Name

Lab Partner 2 Name

CS 115 Fall 2019 Lab #8

Due: **Thursday, October 31st, midnight**

Points: **20**

Instructions:

1. Use this document template to report your answers. Enter all lab partner names at the top of first page.

2. You don't need to finish your lab work during the corresponding lab session.

3. ZIP your lab report and java files (if any) into a single ZIP file. Name the ZIP file as follows:

```
LastName_FirstName_CS115_Lab8_Report.zip
```

4. Submit the final document to Blackboard Assignments section before the due date. No late submissions will be accepted.

5. ALL lab partners need to submit a report, even if it is the same document.

Objectives:

1. (4 points) Demonstrate the understanding of iteration structures.

2. (16 points) Write programs that utilize iteration

Problem 1 [4 points | 1 point each]:

a) Draw a table of i values for each iteration of the loop. Predict the output then verify that your prediction was correct.

```
for (int i = 1; i < 10; i += 2) {
    System.out.print(i + " ");
}
```

EXPECTED OUTCOME: 1 3 5 7 9

b) Draw a table of i values for each iteration of the loop. Predict the output then verify that your prediction was correct.

```
for (int i = 1; i < 10; i++) {
    if (i % 2 == 0) {
        System.out.print(i + " ");
    }
}
```

EXPECTED OUTCOME: 2 4 6 8

c) Draw a table of x, y, and i values for each iteration of the loop. Predict the output then verify that your prediction was correct.

```
double x = 1; double y = 1; int i = 0;
do {
  y = y / 2;
  x = x + y;
  i++;
} while (x < 1.8);
System.out.print(i + " ");
```

EXPECTED OUTCOME: 3

d) Draw a table of x, y, and i values for each iteration of the loop. Predict the output then verify that your prediction was correct.

```
double x = 1; double y = 1; int i = 0;
while (y >= 1.5) {
  x = x / 2;
  y = x + y;
  i++;
}
```

EXPECTED OUTCOME: 0

Problem 2 [7 points]:

Monty Hall Problem - Suppose you are going to be on a game show, and you will be given the choice of three doors:

- Behind one door is a car;
- behind the other two doors are goats.

You pick a door, say No. 1, and the host, who knows what's behind the doors, opens another door, say No. 3, which has a goat. He then says to you, "Do you want to pick door No. 2?"

Is it to your advantage to switch your choice? You could use probability theory to figure out the best strategy, switch or no switch. But instead you decide to write a Java program to simulate millions of plays of the game (such simulation is called Monte Carlo simulation) to help you decide the best strategy.

HINT: Make a method to simulate one trial. The method needs to know which strategy to simulate, switch or no switch. And the method needs to simulate one trial using randomness: What door has the prize? What door did you choose? based on those three items of data (switch or no switch, winning door, door choice), the method should figure out if you win or not and return a value indicating that.

Then call the method many times (maybe a million) for each player strategy (switch or no switch), and output the win percentage for each strategy. To test your program, research the Monty Hall Problem online and see what the expected results should be. Here is a Java program shell to help you get started.

```
public class SwitchNoSwitch {

        public static void main(String[] args) {


        }


        public static int montyHallTrial (boolean playerSwitch){

        }

}
```

SOLUTION:

```
import java.util.Random;   // result should be 66% win if switch,
33% win if no switch

public class SwitchNoSwitch {

    public static Random myRandomGenerator = new Random();

    public static void main(String[] args) {

        int noChangeWin=0, changeWin=0;

        for (int i=1; i<=1000000; i++) {
            changeWin=changeWin+montyHallTrial(true);
            noChangeWin=noChangeWin+montyHallTrial(false);
        }
        System.out.println("Win    percentage    if    no    change:
"+100*noChangeWin/1000000.);
        System.out.println("Win     percentage     if     change:
"+100*changeWin/1000000.);
    }

    public static int montyHallTrial (boolean playerSwitch) {
        int door = myRandomGenerator.nextInt(3)+1;
        int guess = myRandomGenerator.nextInt(3)+1;
        int win=0;
        if  ((door==guess  &&  !playerSwitch)  ||  (door!=guess  &&
playerSwitch)) win=1;
        return win;
    }
}
```

Problem 3 [3 points]:

Write a program (call the class `IntegerSum`) that allows the user to enter integer values repeatedly until the value -1 is entered. When all data input is complete, the sum of all entered integers should be displayed (the value of -1 should not be included in the final sum).

SOLUTION:

```
// Let's import Scanner class to be able to read input
import java.util.Scanner;

public class IntegerSum {

    public static void main (String [] arguments){
        // Declare and initialize the sum variable
        int sum = 0;

        // Declare the input variable
        int input;
        // Let's set up the scanner to read input from keyboard
        Scanner myScanner = new Scanner(System.in);

        // Display prompt
        System.out.println("Enter integer value: ");

        // Get first input value by reading a line from keyboard
        String inputString = myScanner.nextLine();

        // Convert the string to integer using Integer wrapper class
        // What could go wrong here?
        input = Integer.parseInt(inputString);

        // Now we can start iterating (if not -1!)
        while (input != -1){
            // Add current input value to the sum
            sum += input;

            // ...and ask user for another integer
            System.out.println("Enter integer value: ");

            // This is the same process as above, but rolled into one
line
            input = Integer.parseInt(myScanner.nextLine());
        }

        // Display sum
        System.out.println("Sum: " + sum);
    }
}
```

Problem 4 [3 points]:

Write a program (call the class `EvenSum`) that allows the user to enter 10 even integer values and calculate the sum of all numbers. If an odd numbered is entered, keep prompting the user for an even number until it is entered. Display the sum at the end.

SOLUTION:

```
// Let's import Scanner class to be able to read input
import java.util.Scanner;

public class EvenSum {

    public static void main (String [] arguments){
        // Declare and initialize the sum variable
        int sum = 0;

        // Declare the input variable
        int input;

        // Let's set up the scanner to read input from keyboard
        Scanner myScanner = new Scanner(System.in);

        // Now we need to iterate 10 times
        for (int i = 1; i <= 10; i++){
            // ...and ask user for another integer
            System.out.println("Iteration " + i + " Enter even integer
value: ");

            // This is the same process as above, but rolled into one
line
            input = Integer.parseInt(myScanner.nextLine());

            while (input%2 != 0){
                // ...and ask user for another integer
                System.out.println("Iteration " + i + " This was not an
even integer. Try again: ");

                // This is the same process as above, but rolled into
one line
```

```
                input = Integer.parseInt(myScanner.nextLine());
            }

            // Add current input value to the sum
            sum += input;
        }

        // Display sum
        System.out.println("Sum: " + sum);
    }
}
```

Problem 5 [3 points]:

Write a program (call the class `RandomIntegersSum`) that will keep generating and summing integers from the set {1,2,3,4,5,6,7,8,9,10} until:

- The number of random integers generate is equal 10 OR,
- The sum of all random integers generated so far exceeds 40.

Display both the number of integers generated and the sum.

SOLUTION:

```
import java.util.Random;

public class RandomIntegersSum {

    public static void main (String [] arguments){
        // Declare and initialize the sum variable
        int sum = 0;

        // Declare and initialize integer counter
        int numberOfInts = 0;

        // Declare the random integer variable
        int randomInteger;

        // Let's instantiate a random generator object
        Random myRandomNumberGenerator = new Random();


        // Now we need to start iterating, summing, and counting

        do {

            // Generate a new integer using the random number generator

            randomInteger = myRandomNumberGenerator.nextInt(9) + 1;


            // Increase the counter

            numberOfInts++;


            System.out.println("Integer  #:  "  +  numberOfInts  +  "
generated - value: " + randomInteger);


            // Add it to the sum

            sum += randomInteger;



        } while (numberOfInts < 10 & sum <= 40);



        // Display sum

        System.out.println("Number   of   integers   generated:   "   +
numberOfInts + " | Sum: " + sum);

    }

}
```