CISC 110 Lab 4

In this lab, you will write a complete program from scratch. Your program will be a one-player number-guessing game, in which the one player is the computer. The computer has to guess a number within a specified range. You will write code that tells the computer how to guess.

The range and the number will be randomly generated using the Math.random function, described below. Then we'll pretend the computer can only check whether the number is lower, higher than, or equal to its guess; we'll pretend it can't just look at the number. We'll count how many guesses it takes to find the right number.

Here is what you need to do:

1. Try out the Math.random( ) function. The Math.random( ) function does not take any parameters; it returns a pseudo-random number between 0 and 1 (0 <= n < 1, if n is the number returned). So, it could return 0 or 0.333 or 0.5197 or 0.99, etc. Try out the function by calling it within a trace statement.

2. Does it really always return a number between 0 and 1? Write a for loop to try it out. Put your trace statement inside a for loop and iterate 50 times.

3. It's often useful to generate a random integer within a certain range, rather than a real number between 0 and 1. For instance, you might want to generate a random integer that's between 2 and 10.
   Define a new function called randomInteger that has two integer parameters that specify the upper and lower limits of the range you want. Call Math.random( ) inside your function and then convert the result to be an integer within that range. Here is the header line for your function with comments describing what you need to do:

```
/* Generates a random integer within the specified range
 * Parameters:  low  - the low end of the range
 *                    high - the high end of the range
 * Returns:  a random integer >= low and <= high
*/
function randomInteger( low: int, high: int ): int
{
// Call Math.random( ) and multiply the result by high - low
// Then add low to the result and round the whole thing with Math.round
// For instance, if low is 2 and high is 10, multiply the result of Math.random()
// by 8 to get a number between 0 and 8; add 2 for a number between 2 and 10
}
```

4. Modify your for loop from step 2 to call your randomInteger function to test that it works correctly. For instance, try calling it with low = 3 and high = 11. Does it always give a result between 3 and 11?

5. Ok, now you get to start writing the game program. First, call your randomInteger function three times to generate three numbers, which you should store in three variables: a lower limit between 0 and 100, an upper limit between 200 and 1000, and a number to be guessed that is between those two limits. Trace these numbers to see what they are. They will be different every time you run your program. For instance, perhaps one time the lower limit will be 88 and the upper limit will be 487, and perhaps the number to be guessed is 265.

6. We will want to keep track of how many guesses it takes the computer to find the right number, so define a variable called "count" for that purpose. Also, define an integer variable to store the computer's guess.

7. Next make the computer's first guess and set count to 1. A good strategy for guessing is to make a guess halfway between the lower and upper limit. For instance, if the lower limit is 50 and the upper limit is 100, assign guess to be 75: (50 + 100)/2.

8. Next you need to write the most important part of your program: the while loop that repeats until the computer has guessed the number. Inside that loop, include the following steps:
    - Display the guess
    - Check whether the guess is lower or higher than the number
        If the guess is lower than the number, than your next guess needs to be higher, so change the lower limit to be equal to guess + 1
        If the guess is higher than the number, than your next guess needs to be lower, so change the upper limit to be equal to guess – 1
    - Generate the next guess: halfway between the new lower and upper limit
    - Add one to your count of the number of guesses

9. Finally, once the loop finishes, the computer has guessed the number. Display how many guesses it took.

10. Run your program a few times and check the output. Notice how the guessing strategy zeroes in on the correct number. Also notice how many guesses are required each time.