Assignment 2

1. Understanding: Explain the weekly project in English, pseudo code, or drawings.
   1.1. Create a program in C that asks a user to guess a number from 1 to 10 in a few guesses (you decide how many guesses in enough with a loop and variable), says whether a guess is too high or too low each round, and adjusts the boundary of given numbers in a guess based on previous guesses (also, consider when you would use different data types and indicator messages based on the expected questions you would ask the user.
2. Design: describe or draw out how the program should behave
   2.1. Explanation
      2.1.1. initialize the required libraries
      2.1.2. initialize getNum function, which creates a random number
      2.1.3. initialize main function
         2.1.3.1. initialize variables to their base values.
         2.1.3.2. Create the range of numbers from 1 to 10
         2.1.3.3. create a random number and store in num
         2.1.3.4. enter while loop
            2.1.3.4.1. if the input is between the low and high values, increment the guessed variable, otherwise, it is a “free” guess
            2.1.3.4.2. take user input
            2.1.3.4.3. check to see if the input is equivalent to the random number
               2.1.3.4.3.1. break out of the loop
            2.1.3.4.4. check to see if the input is greater than the random number
               2.1.3.4.4.1. decrease the high value to the guessed number and continue the loop
            2.1.3.4.5. check to see if the input is less than the random number
               2.1.3.4.5.1. increase the low value to the guessed number and continue the loop
         2.1.3.5. if the guessed value is greater than or equal to the number of available guesses, tell the user that they ran out of guesses.
3. Testing: design and describe some tests to verify that your code would be working properly. Perhaps create a table (input, expected output, actual output) describing the tests that you plan to perform to demonstrate that your program meets the assignment requirements.
   3.1. There is no single available state at each step along the program path due to the fact that random numbers are in use. I will, however; give a single random value and run through a testing table for it.

Random number	User input	High	Low	output
7	--------	10	1	---------
7	5	10	1	You weren’t quite right :(Too small!
7	8	10	5	You weren’t quite right :(Too large!
7	7	8	5	You guessed correctly!

4. Reflection: now that you are done with your program (even if the program is not complete!) you should discuss the process. You should mention things like:
    4.1. dude, this is me. Of course I understand the problem.
    4.2. Of course all my tests worked out the way that they were supposed to.
    4.3. Dude, this me. Of course it went well.
    4.4. The problem is so simple, i’m not sure what there is to miss.