CS 2213 Advanced Programming Recitation - Exercise

Use random.h interface and random.c implementation – Makefile to compile - Debugger

In this ex, you will implement a new client/driver program using random.h. But before that, let's see how to use random.h and random.c files with the simple hilo.c client/driver program that we discussed in class. Also we will use Makefile.

First follow the "recitation" link in the class web page and download the files under ch03-ex01-random-h directory or simply download ch03-ex01-random-h.tar and untar it.

In linux .tar files are called archive files. A .tar file contains multiple files/directors. For example our .tar file is created using

```
> tar  -cf  ch03-ex01-random-h.tar   ch03-ex01-random-h
```

So, ch03-ex01-random-h.tar contains all the files under ch03-ex01-random-h.

If you download ch03-ex01-random-h.tar and then execute the following command on linux

```
> tar  -xf  ch03-ex01-random-h.tar
```

This will extract all the files and save them under a directory called ch03-ex01-random-h

Now go into that directory (i.e., > cd ch03-ex01-random-h), and execute

```
> make
```

This will compile all the programs for you and create executable files (I assume you added ~korkmaz/cslib into your .cshrc file as we described before). Then you can run hilo program and play the game…

TA will go over the `Makefile` and give some information about make utility.
TA will go over `gdb` and `ddd` and show how to debug a program.

NOW, you are asked to implement another client/driver program using random.h library and provide a Makefile to compile it.

Specifically, your program will allow user to play the 4 digit guessing game that I mentioned in the class. Instead of implementing everything in main() function, you will implement certain parts as functions and call them in your main() function as follows:

Do the followings in main():

1. Call Randomize();
2. Computer generates a 4-digit random number between 1000 and 9999 until all digits are different.
    a. secret = RandomInteger(1000, 9999);
    b. Write a function that finds the digits of secret and stores these digits in an array, say s_a.
    c. Write another function to check if all the digits are different in s_a or not.
    d. If not, go to 1(a).
3. User tries to guess the secret number within 40 attempts:
    a. Ask user to enter a 4-digit number, say ans.
    b. Call the same function in 1(b) to find digits of ans and store them in another array, say a_a
    e. Call the same function in 1(c) to check if all the digits are different in a_a or not.
    c. If not, go to 2(a).
    d. Increase number of attempts.
    e. Compare s_a and a_a to determine the number of digits that are in_place, out_of_place, not_exist.
    f. Print these values, and go to 3(a) until secret==ans or 40 attempts are made.

Compile it using make
You can debug it using gdb or ddd

Here is an example:
Suppose computer generated 4359
If user enters, 3457, your program should say
1 digit is in place
2 digits are out of place
1 digit does not exist

Have fun!

You must submit your work using Blackboard Learn and respect the following rules:

1) All assignments must be submitted as either a zip or tar archive file unless it is a single pdf file.
2) Assignments must include all source code.
3) Assignments must include an output.txt file which demonstrates the final test output run by the student.
4) If your assignment does not run/compile, the output.txt file should include an explanation of what was accomplished, what the error message was that prevented the student from finishing the assignment and what the student BELIEVES to be the underlying cause of the error.
