Arthur Liou
CS362-004

Random Testing Quiz

1) Implement a random tester for the function testme() in testme.c. Your random tester will print the output of any error messages. You should implement inputChar() and inputString() to produce random values.

2) Describe how you developed your random tester in a file called randomstring.pdf.

I started out by creating and building the makefile, to allow my testing and implementation to proceed faster than normal. Then I approached implementing inputChar() by declaring a "char randomchar;" and setting that to a randomly generated char via rand(). From there, I saw that the testme function was looking for a "reset\0" string. So a string of 5 characters and a \0 string ender. Thus, in my inputString(), I created a for loop to generated random strings with length of 5.

From here, I decided to narrow the possible combinations of letters to choose from, so I selected only lower-case alphabet numbers and used inputAlphabetOnly() to grab this (and to generated characters for the inputString() function.

However, with this still in effect, it would take at least a couple million (a sample range was from a few dozen thousand to 25 Million) combinations before running into the error, which comes back with

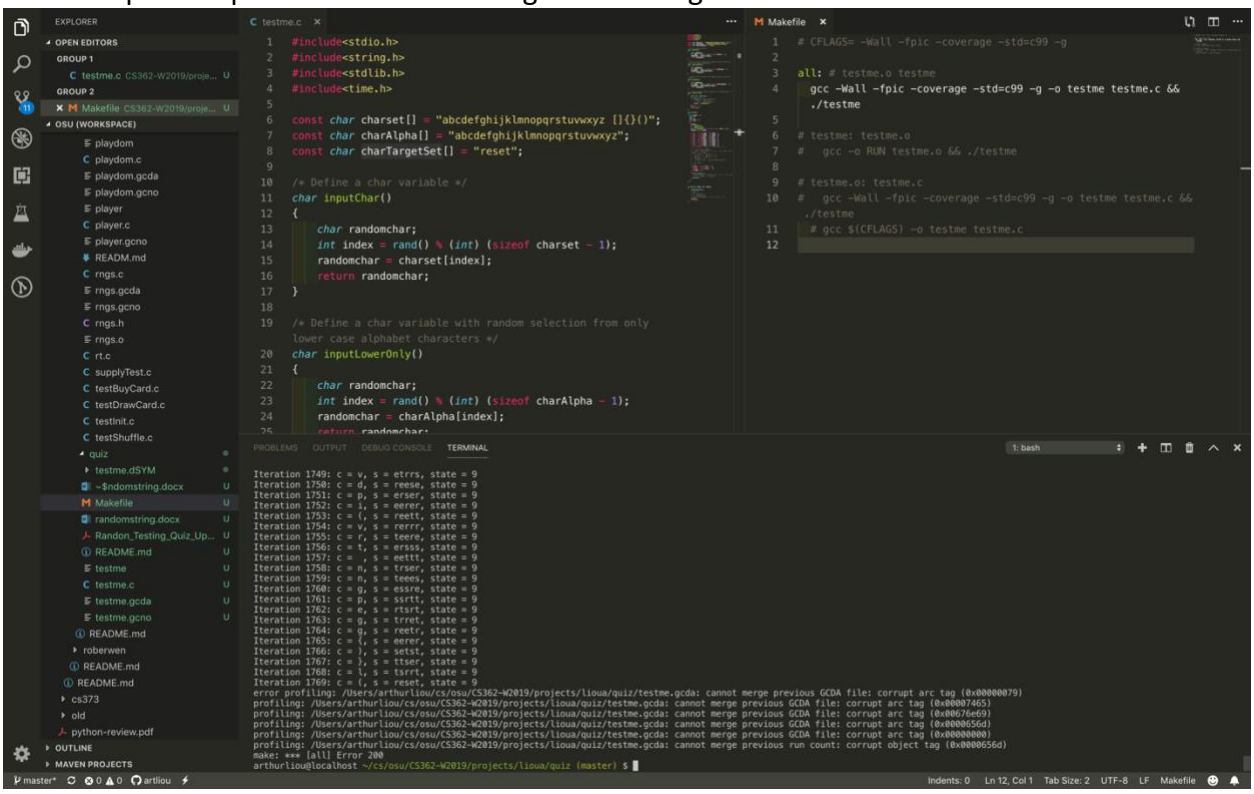
```
"""
profiling: /Users/arthurliou/cs/osu/CS362-W2019/projects/lioua/quiz/testme.gcda: cannot merge previous GCDA file: mismatched number of counters (134)
profiling: /Users/arthurliou/cs/osu/CS362-W2019/projects/lioua/quiz/testme.gcda: cannot merge previous GCDA file: corrupt arc tag (0x00000000)
profiling: /Users/arthurliou/cs/osu/CS362-W2019/projects/lioua/quiz/testme.gcda: cannot merge previous run count: corrupt object tag (0x00000004)
"""
```

The time ranged from a few seconds to 10+ minutes, more often than not landing in the 5+ minute time range.

Then, reviewing the helpful hints: "You can choose and design your input pool (such as, string size (random or fixed), include every character in the ASCII code, include every lowercase letter in the alphabet, or include only the letters used in the target statement, etc.).", I decided to narrow the number of letters I can choose from to just the "letters used in the target statement", so "reset". This would narrow down the random tester to a much shorter duration.

Running the random tester now will return the error, make: *** [all] Error 200, within a minute if not a few seconds, after running through anywhere from ~500-2000+ iterations.

I left my testing code commented out in the testme.c file as well as the ReadMe.md in the Github repo. Sample of the randomstring test running until error below.



3) Create a Makefile and add a rule in the Makefile that will compile and execute the testme.c file.
See attached.

4) Create a branch in your github repository named lioua-random-quiz.
5) Create a directory in this branch (under your onid directory and not under dominion directory), and name it: projects/ lioua /quiz. Submit testme.c and the Makefile in this folder. Check this branch into github.
Done, also present in Master
6) Add a comment in Canvas and give the URL for your fork.

For 4-6)
Branch URL: https://github.com/artliou/CS362-W2019/tree/931989226-random-quiz
Folder URL: https://github.com/artliou/CS362-W2019/tree/931989226-random-quiz/projects/lioua/quiz