

## 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 generate 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 generate 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)
```

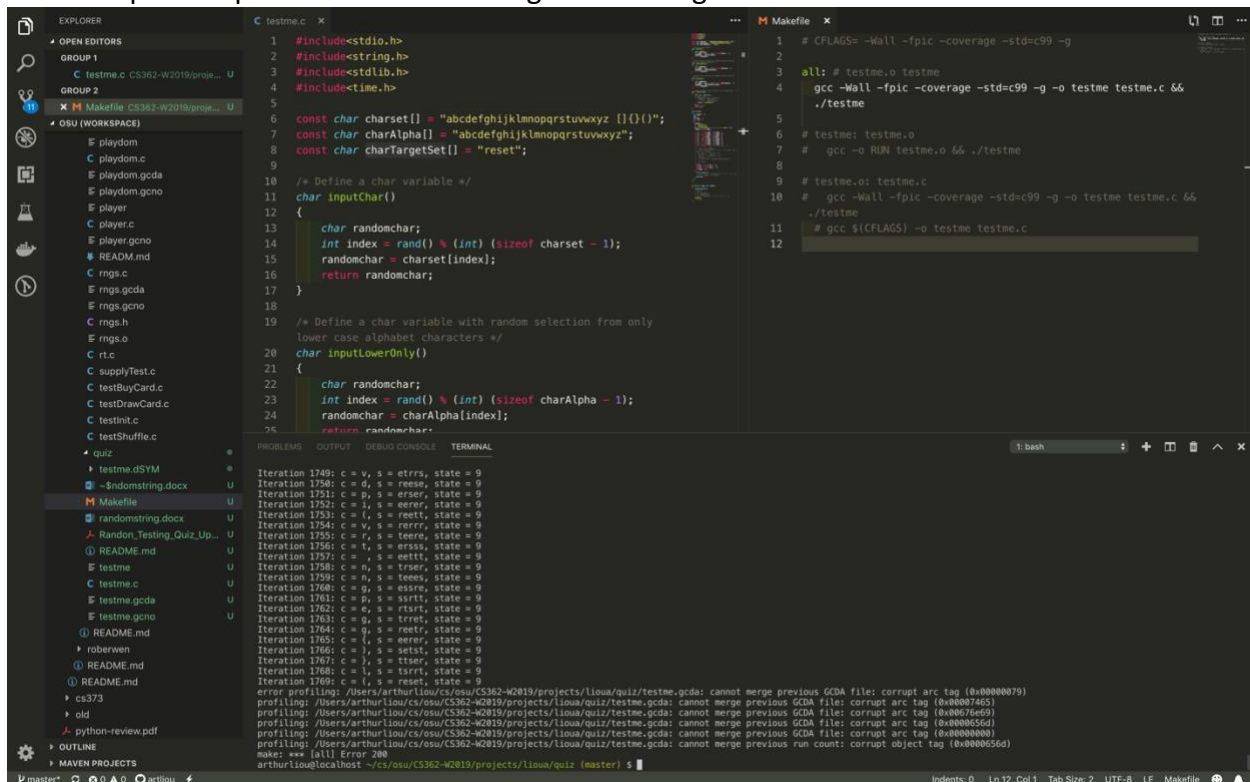
""

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.



```
1 #include<stdio.h>
2 #include<string.h>
3 #include<stdlib.h>
4 #include<time.h>
5
6 const char charset[] = "abcdefghijklmnopqrstuvwxyz {}()";
7 const char charAlpha[] = "abcdefghijklmnopqrstuvwxyz";
8 const char charTargetSet[] = "reset";
9
10 /* Define a char variable */
11 char inputChar()
12 {
13     char randomchar;
14     int index = rand() % (int) (sizeof charset - 1);
15     randomchar = charset[index];
16     return randomchar;
17 }
18
19 /* Define a char variable with random selection from only
20 lower case alphabet characters */
21 char inputLowerOnly()
22 {
23     char randomchar;
24     int index = rand() % (int) (sizeof charAlpha - 1);
25     randomchar = charAlpha[index];
26     return randomchar;
27 }
28
29 int main()
30 {
31     time_t t;
32     srand((unsigned) time(&t));
33
34     /* Test the random string generation */
35     for (int i = 0; i < 200; i++)
36     {
37         char *s = malloc(100);
38         int len = 0;
39         while (len < 100)
40         {
41             s[len] = inputChar();
42             len++;
43         }
44         printf("Iteration %d: c = %s, s = %s, state = %d\n", i, charAlpha, s, state);
45         free(s);
46     }
47
48     /* Test the random string generation with only lower case alphabet characters */
49     for (int i = 0; i < 200; i++)
50     {
51         char *s = malloc(100);
52         int len = 0;
53         while (len < 100)
54         {
55             s[len] = inputLowerOnly();
56             len++;
57         }
58         printf("Iteration %d: c = %s, s = %s, state = %d\n", i, charAlpha, s, state);
59         free(s);
60     }
61
62     return 0;
63 }
```

```
1 # CFLAGS= -Wall -fpic -coverage -std=c99 -g
2
3 all: # testme.o testme
4     gcc -Wall -fpic -coverage -std=c99 -g -o testme testme.o &&
5     ./testme
6
7 # testme: testme.o
8     gcc -o RUN testme.o && ./testme
9
10 # testme.o: testme.c
11     gcc -Wall -fpic -coverage -std=c99 -g -o testme testme.c &&
12     ./testme
13
14 # gcc $(CFLAGS) -o testme testme.c
```

```
Iteration 1740: c = v, s = etrrs, state = 9
Iteration 1750: c = d, s = reese, state = 9
Iteration 1751: c = p, s = erser, state = 9
Iteration 1752: c = j, s = eerer, state = 9
Iteration 1753: c = i, s = reett, state = 9
Iteration 1754: c = v, s = rrrrr, state = 9
Iteration 1755: c = r, s = teere, state = 9
Iteration 1756: c = t, s = ersss, state = 9
Iteration 1757: c = , s = eettt, state = 9
Iteration 1758: c = n, s = trser, state = 9
Iteration 1759: c = n, s = teees, state = 9
Iteration 1760: c = g, s = essre, state = 9
Iteration 1761: c = p, s = srrrt, state = 9
Iteration 1762: c = e, s = rtsrt, state = 9
Iteration 1763: c = g, s = trret, state = 9
Iteration 1764: c = q, s = reetr, state = 9
Iteration 1765: c = i, s = eerer, state = 9
Iteration 1766: c = j, s = setst, state = 9
Iteration 1767: c = j, s = tisor, state = 9
Iteration 1768: c = l, s = tsrrt, state = 9
Iteration 1769: c = i, s = reese, state = 9
error profiling: /Users/arthurliou/cs/osu/CS362-W2019/projects/lioua/quiz/testme.gdca: cannot merge previous GDCA file: corrupt arc tag (0x00000079)
profiling: /Users/arthurliou/cs/osu/CS362-W2019/projects/lioua/quiz/testme.gdca: cannot merge previous GDCA file: corrupt arc tag (0x00007465)
profiling: /Users/arthurliou/cs/osu/CS362-W2019/projects/lioua/quiz/testme.gdca: cannot merge previous GDCA file: corrupt arc tag (0x0075e09)
profiling: /Users/arthurliou/cs/osu/CS362-W2019/projects/lioua/quiz/testme.gdca: cannot merge previous GDCA file: corrupt arc tag (0x000056d)
profiling: /Users/arthurliou/cs/osu/CS362-W2019/projects/lioua/quiz/testme.gdca: cannot merge previous GDCA file: corrupt arc tag (0x00000000)
profiling: /Users/arthurliou/cs/osu/CS362-W2019/projects/lioua/quiz/testme.gdca: cannot merge previous run count: corrupt object tag (0x000056d)
make: *** [all] Error 200
```

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>