Devin Gendron
CS362 - Software Engineering II
Random Testing Quiz
Due: May 12th, 2019

Random Testing Development Process

Before I began writing any code, I reviewed the 'testme' function. It gives a clear indication on how to prepare the tests. I began with the 'inputchar' function. The directions for the random quiz indicate that every ASCII character should be used, so I designed my function for producing characters to do just that. By this point in the OSU program, using the rand() to achieve randomization with a seed for time at the beginning of the program isn't difficult. The next step was to have the range of randomized values we want. Looking at the ASCII table on asciitable.com, I saw that the values I wanted were from decimal 32 (space) all the way to 126 (~). To do this, you set the range that you want the values to be randomized from - usually 0 and some number. You then add to the end after the random number has been picked. Since our range is 32 to 126, I subtracted 32 from 126 to get 94. I then set my range to be from 0-94. Once a random number was pulled from that range, I added 32 to it so that it would always be in the range I wanted. I followed this formula to have the best accuracy:

rand() % (max + 1 - minimum) + minimum

Citation:
https://stackoverflow.com/questions/1202687/how-do-i-get-a-specific-range-of-numbers-from-rand

My function would then return the decimal value which would be classified as a char for its ASCII value.

The next step was the 'inputString()' function. Since we know that the code is supposed to be in all lowercase letters, I simply followed the same procedure with 'inputchar' and randomly selected chars and filled a char array until a combination of 'reset' was found. I had initially set the range to be between 'e' and 't', but I felt it wasn't the appropriate way to randomly test it and set it from 'a' to 'z'. Both implementations worked, with the a-z one taking slightly more time, but very much being within the 5 minute limit (usually under a minute).

Using my makefile with the command “make runtests”, a file randomtests.out is created with the gcov data. The data is:

```
Results for testme.c
File 'testme.c'
Lines executed:97.73% of 44
Branches executed:100.00% of 52
Taken at least once:96.15% of 52
No calls
testme.c:creating 'testme.c.gcov'
```

Here you can see that my line and branch coverage covers what was expected for the assignment.