Ann Kostecki
CS 362-400
7/22/18
Random Testing Quiz

In writing the random tester, I first identified testme() as the function to be tested. I reviewed the function for what variables were being tested and in what way. I found that c (random character), s (random string) and state were being used to eventually return a message.

In my first stage of testing, I removed all instances of randomness in order to guarantee that the testme() function will work if the correct conditions were met. This means that instead of choosing a random character, string and set state, I forced the string to be "reset" and the state to be 9. For the random character, it did not matter what it was when producing the error message. With the string and state set, I was able to return an error message which proved the code worked as intended.

Then, I removed the fixed case and left everything truly random and let testme() run, with the intention of stopping after 5 minutes. However, after just a minute and 2,600,000+ iterations, the program crashed. Upon reviewing the iterations, state quickly changed to 9 but string s was far from "reset".

Next, I considered what values each variable would need to be in order for a change to occur. When testing variable c, ASCII values of 91, 40, 123, 32, 97, 120, 125, 41, and 93 caused a change in state. For variable s, it was values of 114, 101, 115, and 116 that produced the error message. For variable state, it never caused a change to another variable and instead, it was the one to be changed, so its value was mostly unimportant. This means that the range of values that caused a change was from 32-125. This was not a significant change from the 32-127 range already represented in the randomness. Considering this, I decided to focus on limiting the randomness of the string s to be integers/characters between 101 – 116. In order to do this, I modified inputString() to only return characters between the noted ASCII values. After running testme() multiple times, I found that it was returning reset, with the lowest iteration at about 100,000 and the highest at 1,800,000.

With the random selector adjusted to reliably exit testme() (i.e. with an error message), I compiled a test case to run testme() multiple times. Because testme() is void, I was unsure how to check for a return value. My research also could not find how to successfully check the exit status in C. Because of this, I slightly modified testme() to except an integer parameter that would increase if the error/exit conditions were met. This allowed for testing to provide an output for easy review.