Joseph Dutra
dutrajo@oregonstate.edu
Random Testing Quiz
CS362 – 400 – W19
02/09/2019

Random Tester Development:
The first step I took to my development was to really analyze the code section in the testme() function. I realized that the function had an infinite loop that would continue looping forever until a specific set of conditions were met. Each iteration of the while loop will call the inputChar() and inputString() functions and create a new random character or string and then check those random elements against a set of hardcoded conditional statements.

Since the functions we need to create generate values randomly, I needed to figure out what would make an appropriate domain of values for each function. For the inputChar() function, it seemed that the conditional statements would only be met if the character matched one of the expected hardcoded values. To make things simple, I created an array within the inputChar() function and added each character from the conditional statement, and made this array my domain of values to select from. Then it was simple to generate a random integer representing an index within the domain of characters array, and return the randomly selected character.

I then followed a similar process for the inputString() function. When analyzing the conditional for the string portion it became clear that the string was expected to be 6 characters in length since it was looking for the '\0' in index 5 of the string. Then looking at the values it was looking for: 'r','e','s','t' (no need for duplicate 'e' characters since we are just selecting these characters at random. I then generate a simple loop that would randomly select an integer value representing an index into the domain array and thus randomly selecting a character. Once the string was properly built (ending in '\0' character), I return the address to the string in memory for the testme() function to check.

One thing of note, the inputString() function has a memory leak. I tried creating a simple character array and returning a pointer to it, but it kept getting overwritten or changed in the printf statement. I realized that if I malloc'ed dynamically allocated memory for my string and returned that address, then my string wouldn't get manipulated and would properly be checked. The problem is, the function testme() doesn't free the character pointer, so the memory never gets freed. If I was allowed to modify this function, I would add a check to see if the pointer was null, and if not free it.

I was concerned that I made my initial domains too concise, so I tried expanding the number of characters in both functions. All this seemed to do was increase the overall run time of the program. I then saw on the assignment prompt that we could make our random selections as narrowly as we wanted, so I changed my domains back to be as concise as possible to ensure the function ran in the fastest time possible.