Joseph Lewis
Random Testing Quiz

Before I wrote any code, I read the instruction set. This gave me an idea of what I needed to do (implement the inputChar and inputString functions), but it did not tell me anything about the implementation of testme.c. The goal of inputChar was to simply output a random character. The goal of inputString was to output a random string. I was given total control of the length of the string and the value of each character in the string and the single character returned from inputChar.

With this information and no knowledge of the implementation of testme.c, I went on to implement the two functions (black box testing). My approach for each function is described below:

inputChar
Output a random character with a decimal value between 33 and 126. These numbers represent characters on the ascii table that I believed to be reasonable inputs. This one a one line function that called the rand function and performed a modulus operation to get the input in the rand I wanted. In the same line, the integer was cast to a character and returned.

inputString
Allocate a random number of bytes between 1 and 12 (arbitrarily chosen). Then loop over the length of the string and assign each index a random character with a decimal value between 33 and 126 (for the same reason as above). Following the loop, this string is returned. As a result of dynamically allocating memory, free statements were added to testme() to prevent memory leaks.

Running the file as currently constructed resulted in a segmentation fault. This let me know that I wasn't going to be able to write these functions without knowledge of the implementation. I read through the testme function and found that the segfault was due to the string length. The function tests the string's indices up to 5 (length of 6). With this knowledge, I updated the inputString function so the length was always 6.

Running the file now resulted in a very long runtime with poor coverage.

I went back to the drawing board and continued to review the testme function. I realized that it was looking for very specific values of c (the character) and s (the string). So I defined a pair of global strings that only contained the exact characters the conditionals are looking for. I then refactored the functions to randomly select a value from the string related to that function (characters[] for inputChar and stringCharacters[] for inputString).

Now, the program has 100% branch coverage, greater than 95% line coverage and executes within one minute (usually less than 20 seconds).