Aaron Berns
Random Testing Quiz

To develop the solution to this quiz, I first looked at the conditionals in the testme() function to set what characters were needed. I initially wrote the inputChar() function to call the rand() function and return an int between 40 and 125 inclusive, which was then cast to a char and the result could be any character between '(' and '}'. I used main to test this function by having a for loop call and print its return value many times. It wasn't until I attempted the first real call of testme() that I realized I had made a mistake. The testme() output never moved beyond state 3. I looked more closely and say that to move to state 4, a ' ' char was needed, but I had left it out of the range of chars randomly generated. Expanding the range solved the problem.

Next I wrote the inputString() function, which was a frustrating review of string manipulation in c. I got lots of warnings and errors about returning local variables and incorrect casts until I looked at my notes from cs344 and got the right string format using malloc() to allocate a 6 char array. I initially used the same range of values that I used in inputChar() to fill the string with 5 random chars by assigning the random value to the each index from 0 to 5. I had memset all 6 chars previously so the resulting string already had a null terminator. Using main to make several calls to the function led me to believe it was ready.

When testing testme(), after making the change in range described above for inputChar(), the random strings generated were taking a really long time to hit 'reset'. I decided to limit the range to only lowercase letters for inputString() so that the test would complete in a reasonable amount of time, leaving an option for the wider range test. Eventually 'reset' would have been hit with the wider range testing the last conditional of testme() with many more values. The lowercase only test completed in 3,312,461 iterations the first time, but didn't again the second even after 50,000,000 so I again limited the choices to 'r', 'e', 's', and 't'. This lowered the number of iterations, but also led to a missed branch as there weren't enough iterations to allow for each branch to be covered. To allow for a reasonable amount of iterations I lengthened the inputString() string to 10 chars and made the null terminator an option for any index, so the strings could be different lengths up to 9 chars. This led to a test that consistently completed in several thousand or so iterations. As far as coverage, I was able to achieve 97% line coverage and 98% branch coverage as the final conditional leads to a premature exit call and so the branch where the condition is met and execution continues after the conditional never occurs and the function never properly returns.