** Nicolas Sim (simn@oregonstate.edu) **

For the random tester, it was important to look at the actual test so that the inputs could be made randomly at a later time. By inspecting the void testme() function I could see that c was a char and s was a string (or array of chars). The value c had to be either "[({ ax})]" each respectively to the state it was in to move forward to the next state. And at the end the array of chars had to be "reset" in that order in state 9 in order to print the error statement. It is a layered test using if statements that allow a pass if state and (c or s) is in the correct position stated in the if statement. State is used to remember which tests we have passed by incrementation.

My input pool for inputChar was 0 to 127 ascii characters because 256 extended was not needed. Using mod and the number 128 we can find a random char.

My input pool for *inputString which returns an array of characters has an input pool of 10 characters so we loop 5 times to make a random character string to return using mod to calculate a random number 0 to 9, and memset to null all values before we fill our array with random chars. The input pool was "helloreset". The most important part of the input pool was that it included "...reset..." in it in some way. Increasing the input pool in char *inputString() can lead to a longer run time, as well as increasing pool of inputChar().

By using the Makefile and using make, it compiles testme.c and executes it.