You can choose and design your input pool (such as, string size (random or fixed), include every characters in the ASCII code, include every lowercase letter in the alphabet, or include only the letters used in the target statement, etc.).

Your random test generator should not take more than five minutes to achieve the coverage goal.

inputChar():

The first step I took in generating the random test was to look at what the lowest and highest characters needed to be. For writing the inputChar() function, I looked up the corresponding characters in an ascii table, and found that in order to achieve this, I need to generate a number from 32('space' character), to 125('}' character).

When I had the range figured out, I used the rand() function to generate a character in the range, which my function will return back. Here, the line: rand() % 94 +32 means that the generated numbers in the range 0-93 will be upshifted by 32 -- This will make this line return numbers in the range 0+32 up to 93+32, which is, from 32 ('space') up to 125 ('}') inclusive.

inputString():

A similar approach as above was taken for the string function. Looking at the testme.c code, I see that the random string will need to spell 'reset\0'. To make sure the random test generator does not take longer than a few minutes to achieve the coverage goal, I decided to limit the character range for this test.

Using a similar approach as above, my line: rand() % 16 +101 means that I add 101 to the generated numbers in range 0-15 to make them fall in the range of 0+101 to 15+101, which is, from 101 (e) to 116 (t) inclusive. Note that strings will end with a '\0', and so I did not need to generate the null terminator. In accordance with the time limits, I decided to create these random strings of fixed lengths only. For the string, I created an array of size 6 (to hold 5 characters of reset which are generated in a loop, plus the last element will be the null terminator).