Random Testing

In order to perform the assignment I had to alter 2 functions inputChar() and inputString().

The nature of the program is to get a random character and a random string. The The first block of the program function testme() will determine the state number via the character c. If the character c matches those in the program, it will increment the state to a maximum of 9. Once a state of 9 is reached, then the random string really comes into play. The random string must equal “reset” in order to finish and print the error.

inputChar was easy to implement. The range of the ASCII characters was between 125 and 32. And due the task of it, it only had to find a single char, instead of a specific order of chars. In order to get a char, a random number between 32 and 125 was chosen. The number was then cast as a char, and simply returned.

inputString was a lot harder to implement. I first experimented with an array of chars, however I could not get it to return properly and continuously got segmentation faults. In the end I had the function allocate memory for a new string. The method of getting chars to make a string was similar to that in inputChar. However the range for the letters required to exit the program was a lot larger. The function would iterate from 0 to 5 and assign a random char to the allocated string. The last element in the string was always set to a null terminator. At first I was using ASCII character between 101 and 116, however this was way too long and took anywhere between 100k and 1Million iterations before finding the right combinations. In order to greatly shorten the time, I created a pool of 10 chars that the function could choose from. This still could cause anywhere between 100k and 400k, but it was a lot quicker. Another import thing was that I had to free the string after it was used each time it was called or else there would be massive leaks.