

Analysis and Development

Testme() produces the error statement if, and only if, the string `s`, which is generated by the function `inputString()`, spells `reset`, ending in a null terminator, and `state` equals 9. The state begins at 0 and in each iteration of `testme()`'s while loop can be incremented a maximum of one time, if char `c`, which is generated by the function `inputChar()`, matches the correct char for the current state. In summary to produce the error message iterations of the while loop must receive the correct values of `c` in order, but not consecutively, until state 9 is reached, and then `s` must be `reset\0`.

The first thing that occurred to me is that randomizing `c` and each element of `s` across the range of all possible chars would be extremely time consuming to produce the error message. There are 256 values that a char can hold. To progress through the states it would not take too long as each step is a $1/256$ chance, but each step is independent. However, the string `s` requires that 6 chars be in an exact order at the same iteration of the loop. This is 256^6 possibilities which is incredibly large.

Instead, I chose to limit the randomized choices to the lower case English alphabet for `s`. `inputChar()` chooses a random value for the char in each loop. `inputString()` chooses a random lower case letter from `[a,z]`. This allows the for test to check invalid values while still restricting the size of the problem to 26^5 , which is manageable. I manually add the null terminator to the end of the string to prevent C from accessing memory incorrectly. The entire string is then returned. The solution works by randomly choosing values until the right values are chosen in order to progress state from 0 to 9 and then produce the correct string.