Sharaya Baker

October 28, 2018

CS 362 – 400 – F18

Random Testing Quiz

The first step in deciding how to develop tests is determining what the final state needs to be in order to achieve the desired output. In analyzing the final "if" statement, I saw that the final state needed to be 9, and the string of 6 chars needed to be specific values "reset". Seeing that the testme() function is a "while" loop, I determined that this would run until the specified values in the target statement were met. Therefore, I could have the inputChar() and inputString() functions simply return one random char or string and it would run as needed.

char inputChar();

In the first testme() "if" statements, I saw that the chars were in a fairly wide range on the ASCII table. However, I could not determine a way to narrow down this range without totally fixing the chars, which seemed to be not much of a test at all.

Therefore, I used a formula to get random numbers (ASCII Dec) in a specific range (source: https://stackoverflow.com/questions/2509679/how-to-generate-a-random-integer-number-from-within-a-range):

```
r = (rand() % (max + 1 - min)) + min
```

In this case, the min would be the lowest ASCII Dec for the target statement, which is a space (ASCII 32). The max would be the highest ASCII Dec for the target, which is a '}' (ASCII 125).

Knowing that C automatically turns ints into their ASCII equivalent when casted as a char, I simply set the formula equal to a char and returned that char.

char* inputString();

For the random string, I decided that a char array of the necessary size (6) would be the best route since a char* cannot be changed. I then used the same formula in a "for" loop and put random chars into the array in the range of ASCII 101 ('e') to ASCII 116 ('t') at index 'i'. I had the loop iterate only 5 times, rather than the full length of the array (6) because a string must end in a null terminator. I then simply assigned the null terminator to index 5 of the array. This has the added benefit of reducing the number of times rand() will be called, also reducing the time the test will run.

Since a char[] and char* are essentially equivalent in C, I just returned the char[] without casting to a char*. This does throw a warning when compiled, but it has not seemed to cause a problem when running.