CS362 Winter 2019

Random Quiz

Craig Ricker – 933229368

Table of Contents

Overall Design

We are implementing two functions, char inputChar() and char * inputString(). These functions are then utilized in the function testme(). Testme() contains a loop that repeatedly runs containing a switch. Each iteration a single char is generated by inputChar(), and it progresses from case 0 to case 9, and relies on receiving characters in a specific order to successfully progress from case 0 to case 9. Once case 9 is reached, the string that is generated each loop is then considered. If the string is exactly “reset\0”, an error message is printed, and the file exits. This is the entirety of the program. In order to ensure full coverage of the file, we want to access all cases, and ensure that the program successfully reaches state 9, and receives the string “reset\0” successfully within five minutes. Our actual assignment is to implement inputChar() and inputString().

inputChar()

To ensure full coverage, we ultimately need to input the following chars in sequential (but broken up) order: [, (, {, space, a, x, },),]. Once these chars are entered in this order, you will be at case 9, and be able to exit the program once the correct string is generated. For this, I used rand() to generate a single random character between 32, and 126 in the ascii table. These numbers were chosen to ensure that state nine is reached in a reasonable amount of time, but not in a direct manner. With random testing, it is important to not perfectly generate the correct input, you do want to expose it to “wrong” cases that will not perfectly progress the program.

inputString()

For generating a string, it was essential that we limit the choices of random to ensure that it ran within the allotted time. We rapidly get to “state 9” controlled by the char generated by inputChar(), and spend the majority of the time looking for “state\0”. To ensure that the text is

generated within a reasonable time, I restricted the letters to only lowercase, and the correct length. I initially exposed the generator to all asci characters, and variable length of 1-10 characters…but this was MUCH too long.

Lessons Learned:

This assignment was very educational on the power, and the drawbacks of random testing. My first implementation, while correct and ensured coverage IF it had the time to fully run, was terrible because of how slow it was. With white box, random unit testing it is important to ensure you limit the scope of your random to the domain that is important. You have additional insight to the functions that are being tested, and it is important to put that knowledge to good use.