Brandon P. Kyriss

CS 362_400

Quiz 2: Random Testing

When I began designing a test to produce the error message in testme.c, I developed inputChar() to generate a random ASCII character from the set of ASCII printable characters (32 to 126). The conditions for advancing the 'state' of testMe simply require the input of a single target char, allowing the use of all possible char inputs still advances the states to state 9 very rapidly (1 in 128 chance of choosing the correct character to advance the state every iteration).

I initially designed inputString to generate a \0 – terminated 'C string' composed of 5 completely randomly generated ASCII characters in the range 'a – z.' This test obviously does not test the full range of possible ASCII characters that could possibly entered. This led me to thinking about what the purpose of this test might be in a real life testing scenario. I feel like I have a bias in thinking about this test due to the fact that I think like a developer and the test is white box, allowing me to see the code. I technically know the exact input that needs to be entered in order to produce the required error message, so I wanted to make sure I had enough randomness to keep from biasing the test too much . I also don't know what the real world context of this test might be. Is the user entering the string? Is it to be entered programmatically? Will the input be validated to ensure that there are only going to be lowercase letter characters entered, or will it need to deal with all possible ASCII chars. It could be capital and lowercase chars. I decided that I would assume that only lowercase letters would be entered.

Running the test using a completely pseudorandomly-generated string of 5 chars + \0 took a long time to run (50 million+ iterations, taking about 10 minutes or so). Statistically, I wondered how many random strings really needed to be tested to get useful results out of the test. I ended up rewriting inputString() in a way that statistically increased the chances of generating the letters r,e,s, and t, the letters that make up the string 'reset' which throws the required error. This approach ensures that ~200,000 to > 1 million iterations of all possible ASCII characters 'a-z' are tested each run, allowing the test to run much faster for the TAs. I believe that this method retains more randomness than other methods I discussed with others on the class Piazza boards, such as decreasing the range of chars used to only part of the lowercase alphabet.

I increased the chances of hitting choosing the required letters by instantiating an array chars[] containing all of the lowercase letters in the alphabet, but duplicating my target letters so there were 2x as many to be selected from the list.

My test can be compiled with make testme and run with the command testme or ./testme.