Course: CS362
Name: Artem Slivka
Date: 10/26/18

Random Testing Quiz

Git URL:
https://github.com/aslivka/CS362-004-F2018/tree/slivkaa-random-quiz/projects/slivkaa/quiz

Development

inputChar()

After examining testme.c code, inputChar() has to return a random character to testme() function. Function testme() expects the following characters in order from inputChar(): "[({ ax})]". First, I defined a hardcoded string, keyList, in inputChar() that held these wanted characters and b-z also.

```
char keyList[] = " [](){}abcdefghijklmnopqrstuvxyz";
```

Next, the function generated a random integer using rand() in the valid index range of keyList string (0 to string length of KeyList -1). Finally, it returned the character at random index of keyList string back to testme(). After including additional characters(b-z) in string, the program runtime didn't increase by much. An average run takes 1-3 seconds, with 7,000 iterations.

inputString()

A similar process was followed for the inputString() function. First, I defined a string of search characters, searchStr, with contents: "resetRESET". To lengthen program runtime a little, extra invalid characters were added to it: "RESET". After that, a loop ran which selected 5 random characters from the searchStr and assigned them to return string, randStr. After randStr was filled up, it was returned to testme() for verification.

How the Program works

1. Program keeps selecting random characters using inputChar until a valid set of characters have been found. The great thing about this implementation is once a character from valid set has been found, the search starts for the next one instead of completely starting from scratch. If valid characters were expected in order without invalid ones somewhere in the middle, the random search would take much longer.
2. After getting a valid set of characters (after doing enough random selections), program starts searching for valid string instead. In this case, the key string == "reset". Once it finds it, the program finishes.

To run program with makefile: enter "make".
In my experience, the entire program usually runs for only about 2-3 seconds, with 5,000-30,000 runs total.