Random Tester – Quiz

CS362 – 400

Victoria Dorn

dornv@oregonstate.edu

For the two random generation functions, I decided to create a string that contained all of the characters that could randomly be chosen from based on a random index.

For the inputChar() function, all I needed to do was to create a string with the lower case letters, curly brackets, parenthesis and square brackets. These were the characters tested within the testme() function. I went ahead and added the upper case letters just in case that was necessary. Though if there were no time constraint in the testing, I would like to have tested using all ascii characters.

```
//string filled with random chars tested for in the function
char * charOptions = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ{}[]() ";
```

InputChar() uses the random c function to select a character from index 0 to the last index of charOptions (or size of charOptions). The function returns the char variable that was randomly chosen from the charOptions string.

```
//selecting a random char from the charOptions string
char randomChar = charOptions[random() % strlen(charOptions)];
```

For the inputString() function, I chose to again use a string filled with possible characters to be randomly chosen from (like inputChar()). I decided to fill the character array charOptions with random lower case characters only at first. The reason I chose to reduce the charOptions string was the fact that the function was taking too long to run with the same characters as inputChar(). Though, I'm pretty sure this is due to the print statement in the while loop as excessive print statements tend to slow down programs significantly. Even with the reduced character options for inputSting(), I was still having problems meeting the time constraint. To solve this problem, I needed to reduce the set of charOptions once again so that the program would make it to the error statement within testme()in 5 minutes or less. From here, I decided to use only the 4 characters that are tested for in the testme() function for the random string (r, e, s, and t).

```
if (s[0] == 'r' && s[1] == 'e'
    && s[2] == 's' && s[3] == 'e'
    && s[4] == 't' && s[5] == '\0'
    && state == 9){
  printf("error ");
```

The inputString() function only deals with a string of 6 characters because the testme() function only tests the these indices, and since I was already having problems getting the program to run in the 5 minute timeframe, I struck with the minimum length string for the testme() function. For full testing coverage of a function that took random strings, I would want to generate a variable length random

string filled with all available ascii characters, but because of the timing constraint I had to make compromises. The inputString() function also only chooses random characters for index 0-4 (or the first 5 characters) because the last character of a string needs to be a null terminating character ('\0'). I have the function set index 5 to the null terminating character manually else the returned "string" might not actually be a string. It is worth noting here that to fully test a function that takes a string as input, I would need to pass the function strings of random/variable lengths. Sending strings of only one fixed length to a function would most likely not test for all possible bugs in function being tested.

Overall, the new testme.c file is able to find the bug with the use of inputChar() and inputString(). Both functions output random characters by randomly indexing a string filled with the possible characters to test. For inputChar(), this was the lower case alphabet, curly brackets, brackets, and parenthesis. For inputString(), this was the characters r, e, s, and t.

If this were a program that I was assigned to test, I would attempt to talk with the programmer who wrote testme() to see if it would be possible to move the print statement into the error message. We would see be able to see the data needed for the error, but it would allow us to test many more options including:

- Random/variable length strings
- More character options in both inputChar() and inputString()

```
while (1)
{
  tcCount++;
  c = inputChar();
  s = inputString();

  //this print must be removed to speed up the program
  //printf("Iteration %d: c = %c, s = %s, state = %d\n", tcCount, c, s, state);

  if (c == '[' && state == 0) state = 1;
  if (c == '(' && state == 1) state = 2;
  if (c == '{' && state == 2) state = 3;
  if (c == ' ' && state == 3) state = 4;
  if (c == 'a' && state == 4) state = 5;
  if (c == 'x' && state == 5) state = 6;
  if (c == '}' && state == 6) state = 7;
  if (c == ')' && state == 7) state = 8;
  if (c == ']' && state == 8) state = 9;
  if (s[0] == 'r' && s[1] == 'e'
      && s[2] == 's' && s[3] == 'e'
      && s[4] == 't' && s[5] == '\0'
      && state == 9){
    printf("error ");
    //adding print here to state iteration c, s and state value after test completes
    printf("Iteration %d: c = %c, s = %s, state = %d\n", tcCount, c, s, state);
    exit(200);
  }
}
```