Random Tester – Quiz

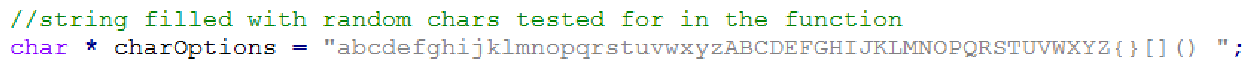
CS362 – 400

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.

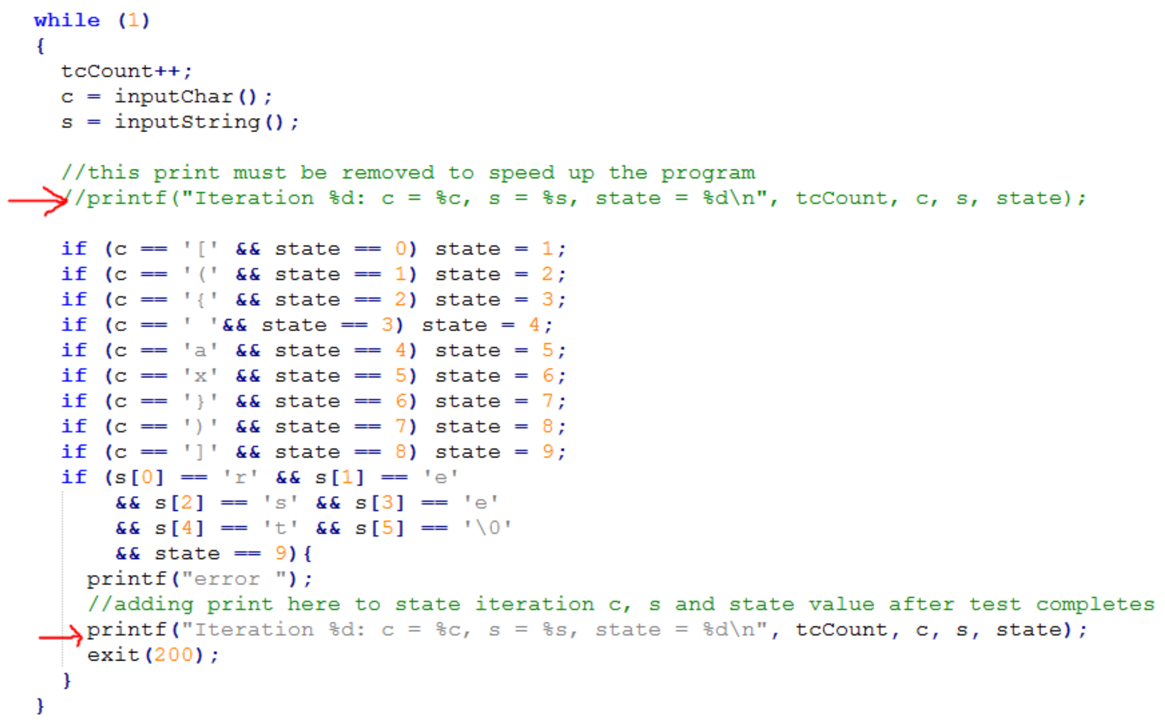
For the inputChar() function, all I needed to do was to create a string with the lower case letters, curly brackets, parenthesis and square brackets. These were the characters tested within the testme() function. I went ahead and added the upper case letters just in case that was necessary.



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

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 array with random lower case characters only. The reason I chose to reduce the charOptions string was the fact that the function was taking too long to run otherwise. Though, I'm pretty sure this is due to the print statement in the while loop. I will discuss this in a bit. The function only deals with a string of 6 characters because the testme() function only tests the these indices. 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. The function also only chooses random characters for index 0-4 because the last character of a string, but this test had a timing constraint in which the test needed to run in 5 minutes or less. 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.

As I was developing this function, I was having a hard time getting the results within the limited time. So, I decided to remove the print statement from within the while loop to see if my logic was at least correct and the function would eventually error. As soon as I removed the print statement from the while loop, the function printed "error" and quit. For this reason, I moved the print statement to directly after the error print. The print now only prints once, but the tester runs much faster.



```
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);
  }
}
```

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. The string being randomly index is filled with characters chosen based on the testing criteria found in the testme() function. Finally, the testme() function needed the print statement moved out of the outer while loop as it was significantly slowing down the program. After the print statement was moved, the program ran within the allotted time (actually much faster than the allotted time).