Random Quiz Writeup

1. **Summary of Process**

   I started developing the input distribution by trying large ranges for the inputs. For the character input, c, I tried all the values in the ASCII set from 0 to 127. I then decided to cut out the characters from null to unit separated (0 to 31) because I did not think they were relevant inputs for such a program. The code very quickly reaches state 9, and on inspection, if the program reaches state 9, then it has reached every state before it, which guarantees states 0 through 8. Morever, on inspection there is no way for the state to change from 9 to any other value. Therefore, the choice of how large to make the random inputs is fairly irrelevant for c.

   The string, s, on the other hand, is harder to generate random inputs while still maintaining full branch coverage. This is because of the following branch in the code:

   ```
   if (s[0] == 'r' && s[1] == 'e'
      && s[2] == 's' && s[3] == 'e'
      && s[4] == 't' && s[5] == '\0'
      && state == 9)
   {
      printf("error ");
      exit(200);
   }
   ```

   In effect, whether this branch is hit in a reasonable amount of time depends on the choice of input. If the input distribution has fixed length strings of five and if there are x possible characters that might fill each spot in the array, then there are x^5 possibilities. Using the full unextended ASCII character set, then, there are $127^5 = 33,038,369,407$ strings that can be formed. Moreover, I noticed that executing this code will execute less than 100 million iterations in under five minutes, so the chance of hitting a one in 33 billion chance is unlikely. So, I restricted the choice of input to the lower-case ASCII characters, which only has $26^5 = 11,881,376$, which is far more reasonable in a small amount of time.

2. **Code**

   The inputChar() function is very simple:

   ```
   char inputChar()
   {
       return rand() % 94 + 32;
   }
   ```

   This line of code will guarantee a random value in the range 32(space) to 126 (tilde), which covers all relevant inputs.

   The inputString() function is similar:

   ```
   char *inputString()
   {
   ```

```
        static char randStr[6];
        randStr[5] = '\0';

        // fill all values with random values
        for (int i = 0; i < 5; i++) {
            randStr[i] = rand() % 25 + 97;
        }
        return randStr;
    }
```

This code will set the sixth char in the static string, randStr, as a null terminator and then it will set every other character to a random value in the range 97(a) - 122(z).