Armand Parajon

ONID: Parajona

Assignment:  Random quiz

The two sources of randomness are in the functions inputChar() and inputString(), which are called upon in the testme() function.

```
void testme()
{
  int tcCount = 0;
  char *s;
  char c;
  int state = 0;
  while (1)
  {
    tcCount++;
    c = inputChar();
    s = inputString();
    printf("Iteration %d: c = %c, s = %s, state = %d\n", tcCount, c, s, state);
```

Through a series of iterations where *state* is determined by the outputs of the two input functions, the goal is to output the error message with an exit number of 200:

```
  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 ");
    exit(200);
```

## inputchar

```
char inputChar()
{
    int randNum = rand() % 9 + 1;
    switch(randNum){
        case 1:
            return '[';
        case 2:
            return '(';
        case 3:
            return '{';
        case 4:
            return ' ';
        case 5:
            return 'a';
        case 6:
            return 'x';
        case 7:
            return '}';
        case 8:
            return ')';
        case 9:
            return ']';
        default:
            return 'b';
    }
}
```

This function uses a random number between 1 and 9 to return a character in the set of tested characters. Though the random number should be between 1 and 9, a default condition is added to the end of the case switch statement to handle the unlikely situation in which it doesn't.

## InputString

```
char *inputString()
{
   char* returnArray = malloc(6 * sizeof(char));
   int* randArray = malloc(5 * sizeof(int));
   int i = 0;
   for (i = 0; i < 5; i++){
       randArray[i] = rand() % 4 + 1;
   }

   for (i = 0; i < 5; i++){
        int randNum = randArray[i];

        if (randNum == 1){
            returnArray[i] = 'r';
        }
        else if (randNum == 2){
            returnArray[i] = 'e';
        }
        else if (randNum == 3){
            returnArray[i] = 's';
        }
        else
        {
            returnArray[i] = 't';
        }
   }

    returnArray[5] = '\0';
    return returnArray;
}
```

This function generates a random string of length 5, containing a permutation of the characters r, e, s, and t. The $6^{th}$ character is set to be the null terminator symbol \0. To create a permutation, a random number array of size 5 is generated, with a range of 1 to 4 since there are only 4 characters in the permutation set. After generation, the random number array is iterated over, and the random string is populated in parallel, with the random number at that index determining which character to add.

**Execution**

Since the state variable is saved (due to its scope outside the while loop), it saves and gets progressively updated as state conditions are met. Inevitably, state 9 and the random string “reset” will be generated, and the error message will print. To run, simply type “make” in terminal, and the script will compile and execute. All iterations will be printed out until the error message is hit. Once done, type “make clean” to get rid of the executable and object files.