Random Tester Documentation

liam beckman

1 Quick Start

To compile and run the testme program, simply enter `make && make run`. This will begin the program iterations. To remove build artifacts, enter `make clean`.

2 Implementations

2.1 char inputChar()

It was found that including all 127 ASCII characters does not result in a significant time restraint. In other words, the `testme` function proceeds through all 9 states in a reasonable time frame, even though the "pool" of available characters includes 127 possibilities.

```
1 int random;
2 char character;
3
4 random = rand() % 127;
5 character = (char) random;
6
7 return character;
```

Listing 1: inputChar() implementation

2.2 char *inputString()

In contrast to `inputChar`, the implementation of the `inputString` function was scaled down to only have available the charcters used in the target statement (e.g. 'r', 'e', 's', 'e', 't', and '\0'). Including all 127 ASCII charcters, or even just lower case ASCII letters resulted in many more iterations and an associated longer wait time.

It should be noted that the inclusion of two 'e' characters reflects the frequency of that character in the target statement.

An informal inspection of the running program yields an approximate average of 12,000 iterations until the target statement is returned.

```
int random;
char *string;
int size = 6;
string = malloc(size * sizeof(char));

int i;
for (i = 0; i < size; i++)
{
    random = rand() % size;
    switch (random)
    {
    case 0:
        string[i] = 'r';
        break;
    case 1:
        string[i] = 'e';
        break;
    case 2:
        string[i] = 's';
        break;
    case 3:
        string[i] = 'e';
        break;
    case 4:
        string[i] = 't';
        break;
    case 5:
        string[i] = '\0';
        break;
    }
}

return string;
```

Listing 2: inputString() implementation