

For All Code, There Exist Properties to be Checked

(or, random testing with FsCheck)

Paulmichael Blasucci, Senior Software Engineer at



twitter.com/pblasucci

github.com/pblasucci

linkedin.com/in/pblasucci

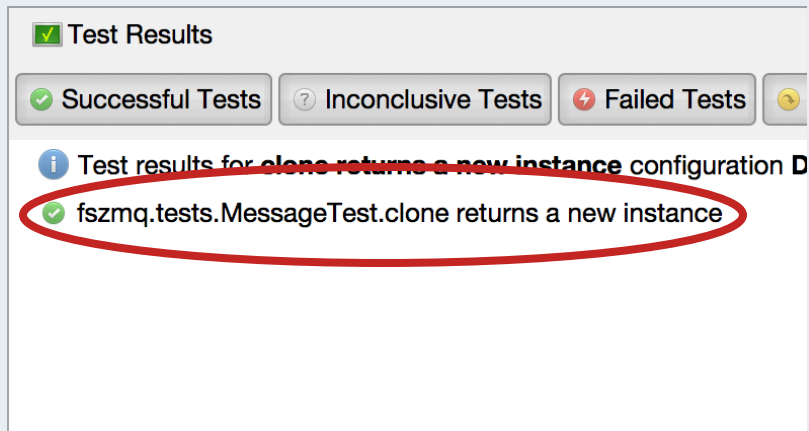
Random Testing

“Properties are described as ... **functions**, and can be **automatically tested on random input...** [or] custom test data generators.”

from [ICFP'00 – Claessen, Hughes](#)

From Unit Testing...

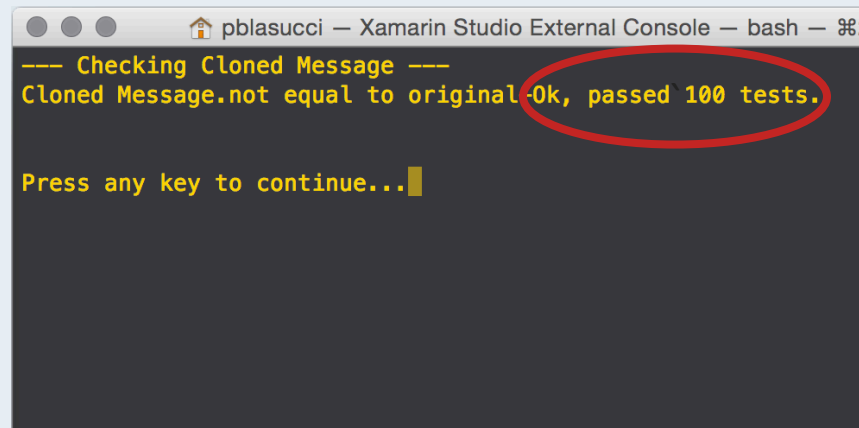
```
[<Test>]
let ``clone returns a new instance`` ()
    use msg1 = new Message("test"B)
    use msg2 = Message.clone msg1
    Assert.That (msg2, Is.Not.EqualTo msg1)
```



The screenshot shows a 'Test Results' window with three buttons: 'Successful Tests', 'Inconclusive Tests', and 'Failed Tests'. Below the buttons, there is an information icon and the text 'Test results for clone returns a new instance configuration D'. A single test result is listed: 'fszmq.tests.MessageTest.clone returns a new instance', which is circled in red.

To Property Testing

```
[<Property>]
let ``not equal to original`` data =
    use msg1 = new Message (data)
    use msg2 = Message.clone msg1
    msg1 <> msg2
```



The screenshot shows a terminal window titled 'pblasucci - Xamarin Studio External Console - bash - ⌘'. The output text is: '--- Checking Cloned Message --- Cloned Message not equal to original -Ok, passed 100 tests.' The text 'Ok, passed 100 tests.' is circled in red. Below the output, it says 'Press any key to continue...|'.

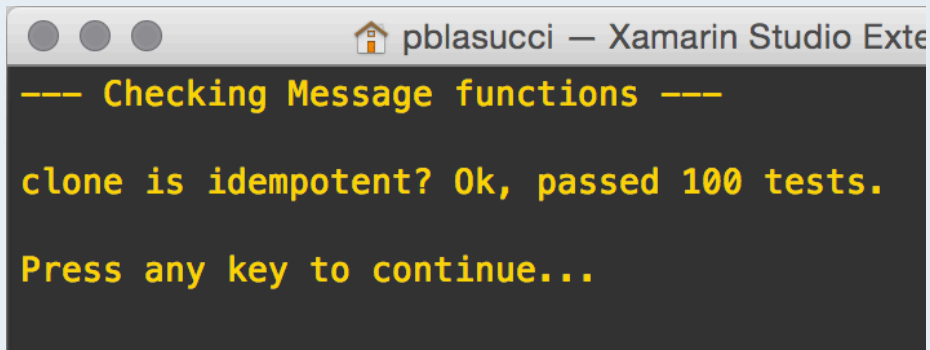
FsCheck: Properties

```
open fszmq
```

```
open fszmq.Message
```

```
let equalContent (msg1:Message) (msg2:Message) =  
    size msg1 = size msg2 && data msg1 = data msg2
```

```
let ``clone is idempotent`` (msg:Message) =  
    use once = clone msg  
    use twice = msg  
        |> clone  
        |> clone  
    twice |> equalContent once
```



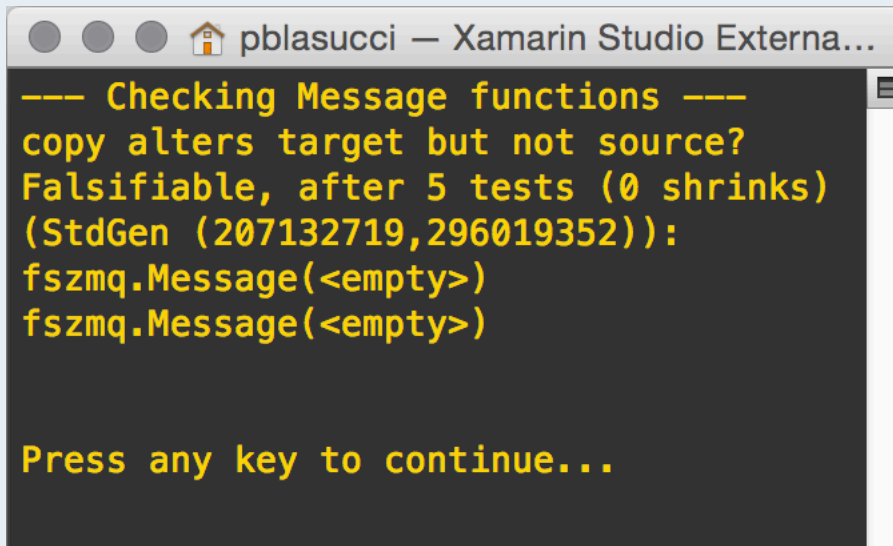
A terminal window titled "pblasucci — Xamarin Studio Ext" displays the following output in yellow text on a dark background:

```
--- Checking Message functions ---  
  
clone is idempotent? Ok, passed 100 tests.  
  
Press any key to continue...
```

FsCheck: Properties

Combining properties (naïvely)

```
let ``copy alters target but not source`` msg1 msg2 =  
  let data1,data2 = data msg1,data msg2  
  copy msg1 msg2  
  (data msg1 = data1) && (data msg2 <> data2)
```



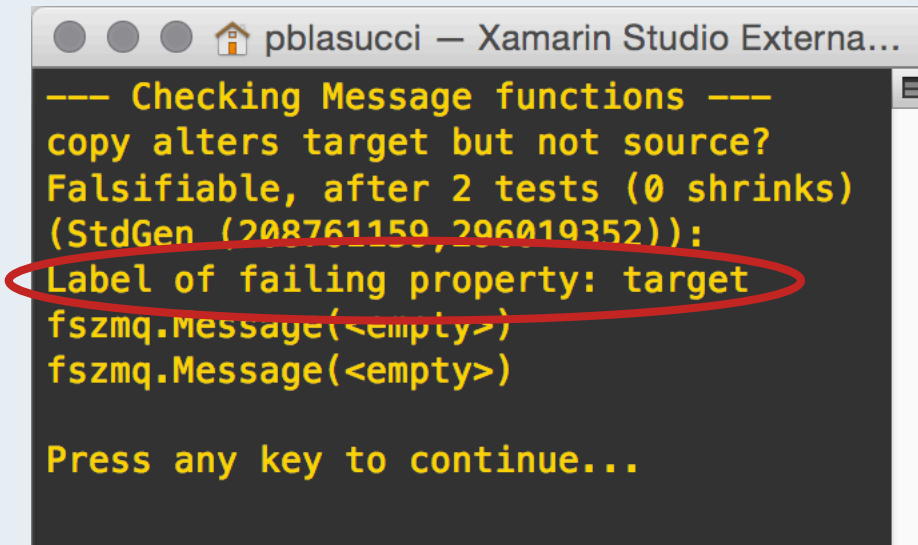
The screenshot shows a terminal window titled "pblasucci - Xamarin Studio Externa...". The terminal output is as follows:

```
--- Checking Message functions ---  
copy alters target but not source?  
Falsifiable, after 5 tests (0 shrinks)  
(StdGen (207132719,296019352)):  
fszmq.Message(<empty>)  
fszmq.Message(<empty>)  
  
Press any key to continue...
```

FsCheck: Properties

Combining and *Labeling* properties

```
let ``copy alters target but not source`` msg1 msg2 =  
  let data1,data2 = data msg1,data msg2  
  copy msg1 msg2  
  (data msg1 = data1 |@ "source")  
  .&.  
  (data msg2 <> data2 |@ "target")
```

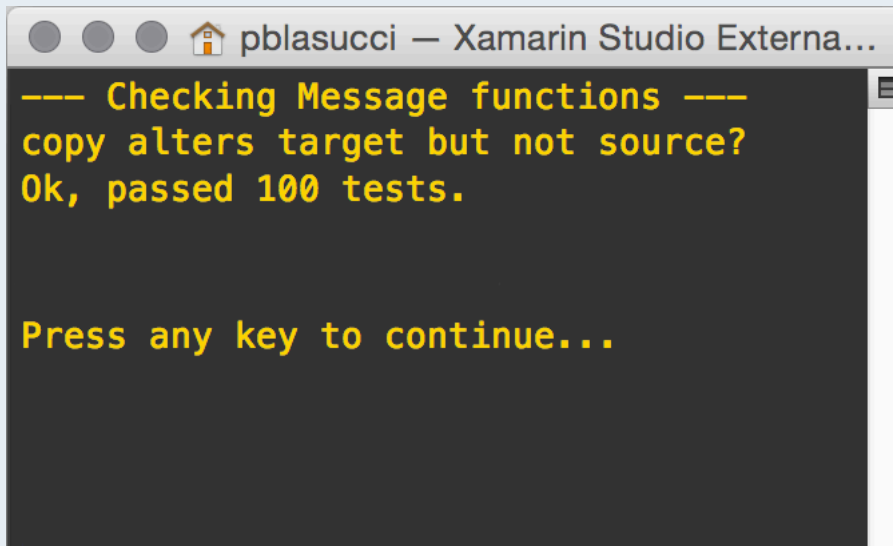


```
pblasucci — Xamarin Studio Externa...  
--- Checking Message functions ---  
copy alters target but not source?  
Falsifiable, after 2 tests (0 shrinks)  
(StdGen (208761159, 296019352)):  
Label of failing property: target  
fszmq.Message(<empty>)  
fszmq.Message(<empty>)  
  
Press any key to continue...
```

FsCheck: Properties

Conditional properties

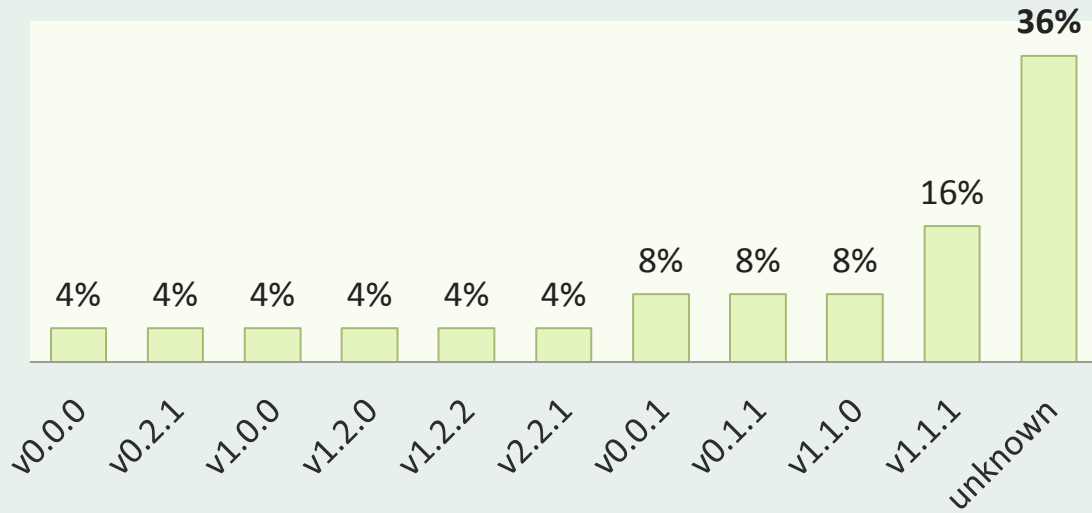
```
let ``copy alters target but not source`` msg1 msg2 =  
  (data msg1 <> data msg2) ==>  
    let data1,data2 = data msg1, data msg2  
      copy msg1 msg2  
      (data msg1 = data1 |@ "source")  
      .&.  
      (data msg2 <> data2 |@ "target")
```



The screenshot shows a terminal window titled "pblasucci - Xamarin Studio Externa...". The terminal output is as follows:

```
--- Checking Message functions ---  
copy alters target but not source?  
Ok, passed 100 tests.  
  
Press any key to continue...
```

Randomly Generated Versions



FsCheck: Generation

distribution of random data using a custom generator

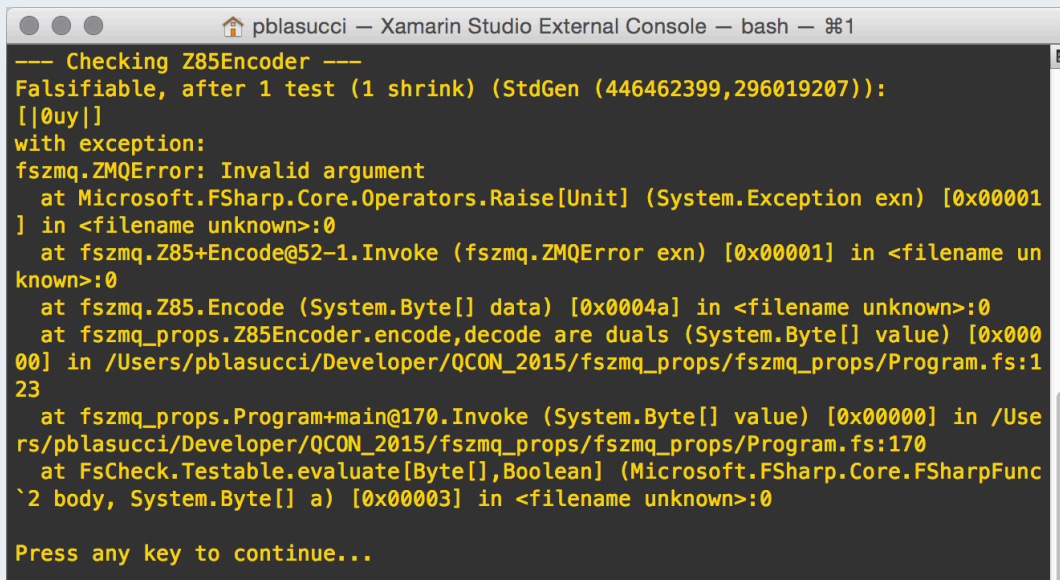
FsCheck: Generation

```
type Generators =
  static member Version =
    let unknown = Gen.constant Unknown
    let version = Arb.generate<NonNegativeInt>
      |> Gen.three
      |> Gen.map (fun (NonNegativeInt m
                      ,NonNegativeInt n
                      ,NonNegativeInt r) -> fszmq.Version (m,n,r))
    [ (1,unknown); (2,version) ]
    |> Gen.frequency
    |> Arb.fromGen
```

FsCheck: Generation

Expressing implicit business rules

```
module Z85Encoder =  
    let ``encode,decode are duals`` (data:byte[]) =  
        data|> Z85.encode |> Z85.decode = data
```



```
pbblasucci — Xamarin Studio External Console — bash — 1  
--- Checking Z85Encoder ---  
Falsifiable, after 1 test (1 shrink) (StdGen (446462399,296019207)):  
[|0uy|]  
with exception:  
fszmq.ZMQError: Invalid argument  
    at Microsoft.FSharp.Core.Operators.Raise[Unit] (System.Exception exn) [0x00001  
] in <filename unknown>:0  
    at fszmq.Z85+Encode@52-1.Invoke (fszmq.ZMQError exn) [0x00001] in <filename un  
known>:0  
    at fszmq.Z85.Encode (System.Byte[] data) [0x0004a] in <filename unknown>:0  
    at fszmq_props.Z85Encoder.encode,decode are duals (System.Byte[] value) [0x00  
00] in /Users/pblasucci/Developer/QCON_2015/fszmq_props/fszmq_props/Program.fs:1  
23  
    at fszmq_props.Program+main@170.Invoke (System.Byte[] value) [0x00000] in /Use  
rs/pblasucci/Developer/QCON_2015/fszmq_props/fszmq_props/Program.fs:170  
    at FsCheck.Testable.evaluate[Byte[],Boolean] (Microsoft.FSharp.Core.FSharpFunc  
'2 body, System.Byte[] a) [0x00003] in <filename unknown>:0  
  
Press any key to continue...
```

FsCheck: Generation

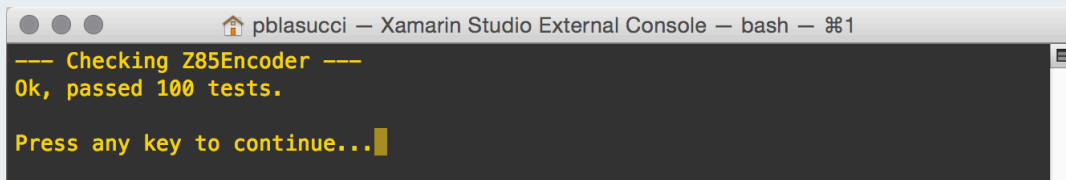
Expressing implicit business rules

“Z85 ... takes a binary frame and encodes it as a printable ASCII string, or takes an ASCII encoded string and decodes it into a binary frame.

The **binary frame** SHALL have a length that is **divisible by 4 with no remainder**. The **string frame** SHALL have a length that is **divisible by 5 with no remainder**.”

from [Z85 RFC](#)

```
module Z85Encoder =  
  let ``encode,decode are duals`` (Mod4Bytes data) =  
    data|> Z85.encode |> Z85.decode = data
```



The screenshot shows a terminal window titled "pblasucci - Xamarin Studio External Console - bash - №1". The output of the test is as follows:

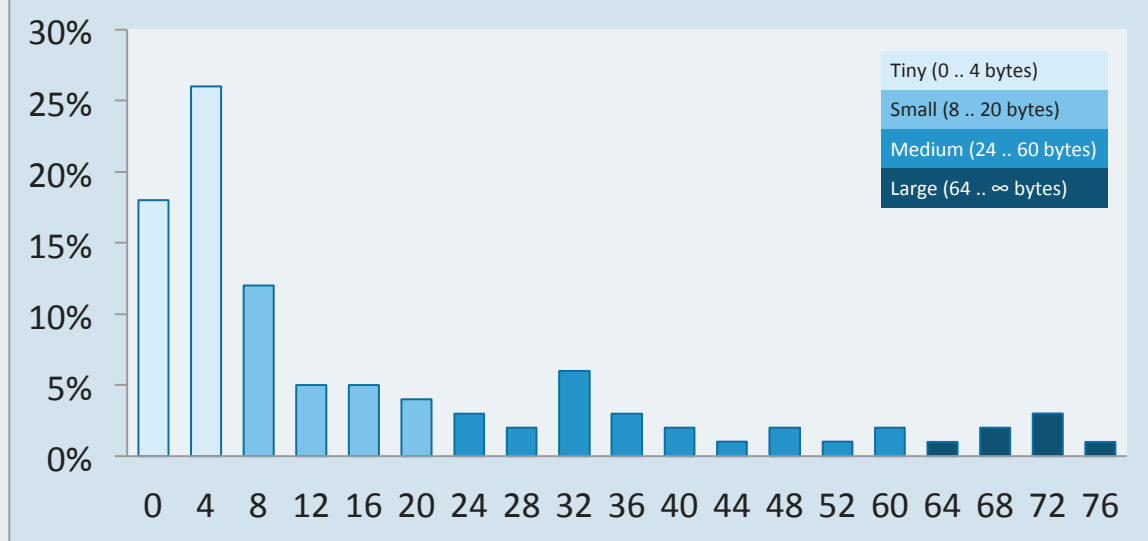
```
--- Checking Z85Encoder ---  
Ok, passed 100 tests.  
Press any key to continue...|
```

```
type Mod4Bytes = Mod4Bytes of byte[]
```

```
// ... elsewhere ...
```

```
static member Mod4Bytes =  
  let g = Arb.generate<byte[]>  
    |> Gen.suchThat (fun b -> b <> null  
                      && b.Length % 4 = 0)  
    |> Gen.map Mod4Bytes  
  let s (Mod4Bytes bytes) = bytes  
    |> Arb.shrink  
    |> Seq.map Mod4Bytes  
  Arb.fromGenShrink (g,s)
```

Observed Array Sizes (in Bytes)



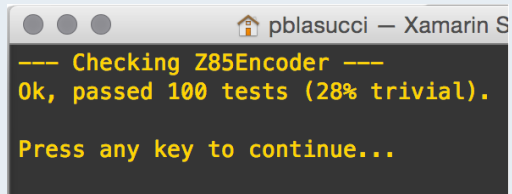
FsCheck: Observations

distribution of randomly generated test inputs

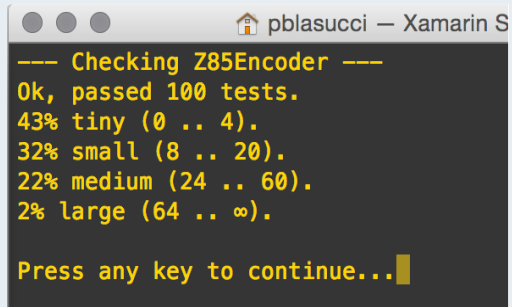
FsCheck: Observations

```
let ``encode,decode are duals`` (Mod4Binary value) =  
  (value |> Z85.encode |> Z85.decode = value)  
  |> Prop.trivial (Array.isEmpty value)
```

```
let ``encode,decode are duals`` (Mod4Binary value) =  
  (value |> Z85.encode |> Z85.decode = value)  
  |> Prop.classify (large value) "large (64 .. ∞)"  
  |> Prop.classify (medium value) "medium (24 .. 60)"  
  |> Prop.classify (small value) "small (8 .. 20)"  
  |> Prop.classify (tiny value) "tiny (0 .. 4)"
```



```
pblasucci — Xamarin S  
--- Checking Z85Encoder ---  
Ok, passed 100 tests (28% trivial).  
Press any key to continue...
```



```
pblasucci — Xamarin S  
--- Checking Z85Encoder ---  
Ok, passed 100 tests.  
43% tiny (0 .. 4).  
32% small (8 .. 20).  
22% medium (24 .. 60).  
2% large (64 .. ∞).  
Press any key to continue...
```

FsCheck: Observations

```
let ``encode,decode are duals`` (Mod4Binary value) =  
  (value |> Z85.encode |> Z85.decode = value)  
  |> Prop.collect (Array.length value)
```

```
let ``encode,decode are duals`` (Mod4Binary value) =  
  (value |> Z85.encode |> Z85.decode = value)  
  |> Prop.trivial (Array.isEmpty value)  
  |> Prop.classify (large value) "large (64 .. ∞)"  
  |> Prop.classify (medium value) "medium (24 .. 60)"  
  |> Prop.classify (small value) "small (8 .. 20)"  
  |> Prop.classify (tiny value) "tiny (0 .. 4)"  
  |> Prop.collect (Array.length value)
```

```
pblasucci - Xamarin S  
--- Checking Z85Encoder ---  
Ok, passed 100 tests.  
26% 4, tiny (0 .. 4).  
18% 0, tiny (0 .. 4), trivial.  
12% 8, small (8 .. 20).  
6% 32, medium (24 .. 60).  
5% 16, small (8 .. 20).  
5% 12, small (8 .. 20).  
4% 20, small (8 .. 20).  
3% 72, large (64 .. ∞).  
3% 36, medium (24 .. 60).  
3% 24, medium (24 .. 60).  
2% 68, large (64 .. ∞).  
2% 60, medium (24 .. 60).  
2% 48, medium (24 .. 60).  
2% 40, medium (24 .. 60).  
2% 28, medium (24 .. 60).  
1% 76, large (64 .. ∞).  
1% 64, large (64 .. ∞).  
1% 52, medium (24 .. 60).  
1% 44, medium (24 .. 60).  
Press any key to continue... █
```

Random Testing

“One of the major advantages... is that it **encourages** us to formulate **formal specifications**, **thus improving** our **understanding...**”

from ICFP'00 – Claessen, Hughes

More Information

about F# and FsCheck

- fsharp.org
- fscheck.github.io/fscheck
- fsharpforfunandprofit.com
- meetup.com/nyc-fsharp

about Paulmichael Blasucci

- twitter.com/pblasucci
- github.com/pblasucci
- linkedin.com/in/pblasucci
- pblasucci.wordpress.com