MUTATION TESTING

```
exM :: Integer -> Integer -> Integer -> Integer

exM _ _ 0 = error "division by zero"

exM x 0 m = (x^0) `mod` m

exM x 1 m = x `mod` m

exM x y m |                 error "negative exponent"

exM x y m | (y `mod` 2) ==0 = (exM x (y `div` 2) m)^2                m

          |                 | otherwise = (exM x (y-1) m) * x `mod` m
```

Lots of mutants and killing!

How it works?

```haskell
crappyHspecTest :: IO ()
crappyHspecTest = hspec $ do
    describe "Unit testing of validity" $ do
        it "Tries a big number to modulate" $ do
            let testNumber = exM 2097152 262144 314
            let reference = ((2097152^262144) `mod` 314)
            testNumber ==reference `shouldBe` (True::Bool)

crappyQuickCheckTest =
    quickCheck (\(Positive x) (Positive z) -> (exM x 0 z)==1 `mod` z)
```

Before mutation

```haskell
exM :: Integer -> Integer -> Integer -> Integer
exM x 1 m = x `mod` m
exM _ 0 m = 1 `mod` m
exM 0 _ m = 0
exM x y m | (y `mod` 2) ==0 = (exM x (y `div` 2) m)^2 `mod` m
          | otherwise = (exM x (y-1) m) * x `mod` m
```

```
*Main Test.QuickCheck> crappyHspecTest

Unit testing of validity
  - Tries a big number to modulate

Finished in 0.1690 seconds
1 example, 0 failures
*Main Test.QuickCheck> crappyQuickCheckTest
+++ OK, passed 100 tests.
*Main Test.QuickCheck>
```

```haskell
exM :: Integer -> Integer -> Integer -> Integer
exM x 1 m = x `mod` m
exM 0 _ m = 0
exM _ 0 m = 1 `mod` m
exM x y m | (y `mod` 2) ==0 = (exM x (y `div` 2) m)^2 `mod` m
          | otherwise = (exM x (y-1) m) + x `mod` m
```

After mutation?

MuCheck

MutPY

Pavol Kincel