Assignment 1 solutions

```haskell
import Control.Monad.State -- Needed only for test code
```

1. Rewrite the factorial function I defined above to use `if-then-else` instead of the pattern guard.

```haskell
factIf n =
  if n <= 0 then 1
  else n * factIf (n-1)
```

2. Define this mathematical function using Haskell, name it `quadratic`, and test it.

$$q(a,b,c) = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

```haskell
quadratic a b c =
  (-b + sqrt(b^2 - 4*a*c))/(2*a)
```

3. Define a function to calculate the volume of a sphere of a given radius. Name it `sphere`.

```haskell
sphere r = (4/3) * pi * r^3
```

4. Here is a recursive function in Haskell:

```haskell
slow x y
  | x < y = x
  | otherwise = slow (x-y) y
```

Trace out the steps involved in evaluating `slow 101 21`, the same way we would for an algebraic expression. That is, determine the parameters `x` and `y`, check which case applies, then rewrite it.

- `slow 101 21` Substitute 101 for `x`, 21 for `y`. `x < y` is false, so use the `otherwise` case.
- ⇒ `slow (101-21) 21` Subtract
- ⇒ `slow 80 21` Substitute again, still otherwise
- ⇒ `slow (80-21) 21` Subtract
- ⇒ `slow 59 21` Substitute again, still otherwise
- ⇒ `slow (59-21) 21` Subtract
- ⇒ `slow 38 21` Substitute, still otherwise

- ⇒ `slow (38-21) 21` Subtract
- ⇒ `slow 17 21` Substitute, this time `x < y`
- ⇒ 17 So the answer is x.

5. Write a recursive Haskell function `collatzCount n` that calculates the number of steps it takes to get from n down to 1.

```
collatzCount n
  | n <= 1 = 0
  | otherwise = 1 + collatzCount (collatz2 n)

collatz2 n | even n = n `div` 2
collatz2 n = 3*n + 1
```

Test driver

```
main = flip execStateT (0,0) $ do
  verify "fact 5"       120 $ factIf  5
  verify "fact 6"       720 $ factIf  6
  verify "fact10"   3628800 $ factIf 10
  verify "fact12" 479001600 $ factIf 12
  verifyF "quad 2 9 3"  (-0.36254139118231254) $ quadratic 2 9 3
  verifyF "quad 1 -1 -1"  1.618033988749895    $ quadratic 1 (-1) (-1)
  verifyF "quad 2 8 6"  (-1.0)                 $ quadratic 2 8 6
  verifyF "quad 1 30  " (-0.1050878704703262)  $ quadratic 1 30 pi
  verifyF "sphere 1"      4.1887902047863905   $ sphere 1
  verifyF "sphere 3.75" 220.8932334555323      $ sphere 3.75
  verifyF "sphere 18.1" 24838.441017720263     $ sphere 18.1
  verifyF "sphere 21.2" 39349.20615723923      $ sphere 21.1
  verify "collatz  27" 111 $ collatzCount   27
  verify "collatz  99"  25 $ collatzCount   99
  verify "collatz1024"  10 $ collatzCount 1024
  verify "collatz 118"  33 $ collatzCount  118
  get >>= say . show

  where
    say = liftIO . putStrLn
    correct (k, n) = (k+1, n+1)
    incorrect (k, n) = (k, n+1)
    verify = verify' (==)
    verifyF = verify' (\x y -> abs(x-y) < 0.00001)
    verify' :: (Show a) => (a -> a -> Bool) -> String -> a -> a -> StateT (Int,Int) IO ()
    verify' eq tag expected actual
      | eq expected actual = do
```

```
      modify correct
      say $ " OK " ++ tag
  | otherwise = do
      modify incorrect
      say $ "ERR " ++ tag ++ ": expected " ++ show expected
        ++ " got " ++ show actual
```