Haskell Exercises

Oliver Ray

Week 11

This worksheet provides some additional practice using Haskell.

1. Evaluate the following queries in `ghci`:

   (i) `1+1` (ii) `sin(pi)^2+cos(pi)^2` (iii) `mod 34 5` (iv) `div 34 5`

   (v) `divMod 34 5` (vi) `34 ‘divMod‘ 5` (vii) `1==1` (viii) `(==) 1 1`

   (ix) `1==1 || 2<=3 && not (4>5)` (x) `"Hello "++"World"`

2. Declare a function `allSame` which takes three `Int`'s and returns a `Bool` Define this function so it returns `True` iff all its inputs are all the same.

3. Declare a function `allDiff` which takes three `Int`'s and returns a `Bool` Define this function so it returns `True` iff all its inputs are all different.

4. Declare a function `numVals` which takes three `Int`'s and returns an `Int` Define this function so it returns the number of different input values: i.e. `numVals a b c` returns $|\{\mathtt{a}, \mathtt{b}, \mathtt{c}\}|$ so `numVals 9 5 5` returns 2.

5. Redefine `allSame` and `allDiff` so that they exploit `numVals`.

6. Haskell has a built-in Boolean operator `||` for (inclusive) *or*. Define your own operator `##` for (exclusive) *xor*.

7. Evaluate the query `True ## True`.

8. Define a list reversal using a single application of `foldr`.

9. Define a list append using a single application of `foldr`.

ANSWERS

1.
```
*Main> 1+1 ==> 2
*Main> sin(pi)^2 + cos(pi)^2 ==> 1.0
*Main> mod 34 5 ==> 4
*Main> div 34 5 ==> 6
*Main> divMod 34 5 ==> (6,4)
*Main> 34 `divMod` 5 ==> (6,4)
*Main> 1==1 ==> True
*Main> (==) 1 1 ==> True
*Main> 1==1 || 2<=3 && not (4>5) ==> True
*Main> "Hello "++"World" ==> "Hello World!"
```

2.
```
allSame :: Int -> Int -> Int -> Bool
allSame x y z = x==y && y==z
```

3.
```
allDiff :: Int -> Int -> Int -> Bool
allDiff x y z = x/=y && y/=z && x/=z
```

4.
```
numVals :: Int -> Int -> Int -> Int
numVals x y z
  | x==y && y==z           = 1
  | x==y || y==z || x==y   = 2
  | otherwise              = 3
```

5.
```
allSame x y z  = (numVals x y z)==1
allDiff x y z  = (numVals x y z)==3
```

6.
```
a ## b  = not (a==b)
```

7.
```
False
```

8.
```
rev = foldr (\x xs -> xs ++ [x]) []
```

9.
```
app xs ys = foldr (:) ys xs
```