HW 7 **Prof. Caldwell**
Due: 16 September 2010 **COSC 3015**

1

Problem 1.1. Read chapter 3 of Bird.

Problem 1.2. Do exercise 3.3.1 on pp. 74
Recall the `foldn` function from Bird and presented in class.

```
foldn :: (a -> a) -> a -> Nat -> a
foldn h c Zero = c
foldn h c (Succ n) = h (foldn h c n)
```

Here is the definition of the `Nat` datatype and some functions.

```
data Nat = Zero | Succ Nat
   deriving (Eq,Ord,Show)

nat2int :: Nat -> Int
nat2int Zero = 0
nat2int (Succ n) = nat2int n + 1

int2nat :: Int -> Nat
int2nat 0 = Zero
int2nat (k + 1) = Succ ( int2nat k)

shownat Zero = "Zero"
shownat (Succ k) = "Succ(" ++ shownat k ++ ")"
```

Here are defintions for an identity function, addition, multiplication and exponentiation on `Nat`, all defined interms of `foldn`.

```
id_nat n = foldn Succ Zero n
plus n m = foldn Succ n m
times n m = foldn (plus n) Zero m
expt n m = foldn (times n) (Succ Zero) m
```

Problem 1.3.

a.) Implement a function `nat2int1` using `foldn` that behaves just like `nat2int`.

b.) Implement a function `shownat1` using `foldn` that behaves just like `shownat`. You may find the following function useful in your implementation.

```
wrap left right s = left ++ s ++ right
```

c.) Why can't you implement `int2nat` using `foldn`?