HW 10 **Prof. Caldwell**
Due: 9 October 2011 **COSC 3015**

1 Sets

In class we showed how to model a data-type of sets using lists and instantiating the type as an instance of the `Eq` and `Show` type classes. Here's the code we developed in class with some additional functionality added.

```
module Set where

data Set a = Set [a]

subset (Set xs) (Set ys) = null (filter (notin ys) xs)
  where notin xs x = not (x ‘Prelude.elem‘ xs)

instance  (Eq a ) => Eq (Set a) where
   xs == ys =  xs ‘subset‘ ys && ys ‘subset‘ xs

instance (Eq a, Show a) => Show (Set a) where
  show (Set xs) = fix $ show  $unique xs
    where fix xs = ’{’ : (take (length xs -2) (drop 1 xs) ++ "}")
          unique [] = []
          unique (x:xs) = x: unique (filter (/=x) xs)

empty = Set []

insert x (Set xs) = Set (x:xs)

elem x (Set xs) = x ‘Prelude.elem‘ xs

union (Set xs) (Set ys) = Set (xs ++ ys)

remove x (Set []) = (Set [])
remove x (Set (y:ys)) = if x == y then (Set ys) else insert y (remove x (Set ys))
```

Recall that for sets, order of the elements and the multiplicity (how many times they occur in the list) are not significant. This is reflected in set equality - two sets are equal iff they contain the same elements. A *Bag* or *Multiset* is a structure there the order of the elements is not significant but their multiplicities are. We can characterize equality for multisets if we have a function that counts the number of times an element occurs in the mutiset. For $m, n :: \mathit{Bag}\ a$ and a function $\mathit{count} :: a \rightarrow (\mathit{Bag}\, a) \rightarrow \mathit{Int}$ that counts the number of ioccurences of an element in a bag, we define equality as follows:

$$m == n \overset{\text{def}}{=} \forall x :: a.\ \mathit{count}\, x\, m == \mathit{count}\, x\, n$$

Note that for bags m and n, m is a subbag of n if the following holds:

$$m \subseteq n \overset{\text{def}}{=} \forall x :: a.\ \mathit{count}\, x\, m \leq \mathit{count}\, x\, n$$

For bags m and n you could prove the following theorem holds:

$$m == n \Leftrightarrow m \subseteq n \wedge n \subseteq m$$

This definition provides an alternative path to implementing equality for bags.

Exercise 1.1. Use the code above as a template to implement a data-type of bags. You should instantiate the type as an instace of the `Eq` and `Show` type classes. For the `show` function I like a display the looks like "|[1,2,3]|" for "Bag [1,2,3]" - but maybe you have a better idea.

You should also implement the following functions and constants:

```
subbag :: (Eq a) => Bag a -> Bag a -> Bool
empty :: Bag a
insert :: a -> Bag a -> Bag a
elem :: (Eq a) => a -> Bag a -> Bool
multiplicity :: (Num t, Eq a) => a -> Bag a -> t
union :: Bag t -> Bag t -> Bag t
remove :: (Eq a) => a -> Bag a -> Bag a
```