HW 16 **Prof. Caldwell**
Due: 13 November 2008 **COSC 3015**

Here is the code for an implementation of sets as lists.

```
module Set where

import Lists

data (Eq a, Show a) => Set a = MkSet [a]

instance (Eq a, Show a ) => Show (Set a) where
   show m = "{" ++ contents ++ "}"
     where MkSet m' = m
           contents = reverse (drop 1 (reverse ( drop 1 (show (unique m')))))

instance (Eq a, Show a ) => Eq (Set a) where
    s == t = all ((flip elem) s') t' &&  all ((flip elem) t') s'
               where MkSet s' = s
                     MkSet t' = t

insert x (MkSet m) = MkSet (x:m)
delete x (MkSet m) = MkSet (remove_all x m)
union (MkSet m) (MkSet n) = MkSet (m ++ n)
intersection (MkSet m) (MkSet n) = MkSet (filter ((flip elem) n) m)
ismem x (MkSet m) = elem x m
```

Here is a module containing the supporting definitions for operations on lists.

```
module Lists where

unique [] = []
unique (h:t) = if (elem h t) then (unique t ) else h:(unique t)

remove_all x = filter (/=x)
```

0.1 Multisets

Multisets (sometimes called bags) are like sets except that multiplicity does count *e.g.*

```
[1,2,2] /= [1,2]
[1,2,2] == [2,1,2] == [2,2,1]
```

So the order of elements does not count for determining equality but the number of occurences of each element does.

Definition 0.1 (multiset intersection) For multisets M and N, if there are k occurences of x in M and there are j occurences of x in N then there are $min(j,k)$ occurences of x in the multiset (*m* `` `intersection` `` *n*).

Definition 0.2 (multiset union) For multisets m and n, if there are k occurences of x in m and there are j occurences of x in n then there are $max(j, k)$ occurences of x in the multiset `(`m `` `union` `` n`)`.

Definition 0.3 (multiset delete) Given a multiset m, if x occurs k times in m then there are $max(0, k-1)$ occurences of x in the multiset `(delete` x m`)`.

Definition 0.4 (multiset insert) Given a multiset m, if x occurs k times in m then there are $k+1$ occurences of x in the multiset `(insert` x m`)`.

Definition 0.5 (multiset ismem) Given a multiset m, if x occurs 1 or more times in m then `(ismem` x m`)` is true and is false otherwise.

Exercise 0.1. Write Haskell code to implement Multisets as lists by modifying the Set module presented above. You'll have to reimplement `show`, `==`, `insert`, `delete`, `union`, `intersection`, and `ismem` so they give the proper answers for multisets.