HW 17 **Prof. Caldwell**
Due: 2 November 2010 **COSC 3015**

Exercise 0.1. Read pp. 251-263 of Bird.

In class we discussed abstract datatypes in Haskell and houw to define them using the module system. Here are some links to more about the module system.

http://www.haskell.org/haskellwiki/Abstract_data_type

http://www.haskell.org/tutorial/modules.html

Exercise 0.2. Implement an module for Rose trees having the an abstract type `Rose a` that supports the following operations:

```
leaf  :: a -> Rose a
value :: Rose a -> a
addChild :: Rose a -> Rose a -> Rose a
children :: Rose a -> [Rose a]
foldRose :: (a -> [b] -> b) -> Rose a -> b
```

The following axioms should hold for your implementation:

```
value (leaf x) = x
value (addChild r t) = value t
children (leaf x) = []
children (addChild r t) = r: (children t)
foldRose f (leaf x) = f x []
foldRose f t = f (value t) (map (foldRose f) (children t))
```