HW 12 **Prof. Caldwell**
Due: 30 October 2012 **COSC 3015**

Here is some code for a calculator language similar to the one presented by Hadi in class Tuesday extended to include additional constructors - a `Let` statement, `if-then-else`, `FF`, `Not`, `And`, and `Or`.

In the evaluator[1], rather than include Boolean values, you will interpret false `FF` as 0 and and any non-zero value as true. So, in the context of the Boolean slot in an `if-then-else` expression, something like `Const 19` is interpreted as true and `Const 0` is interpreted as false.

```
data BinOp = Plus | Minus | Times | Div | And | Or  deriving Show

meaning Plus = (+)
meaning Minus x y =  x - y
meaning Times = (*)
meaning Div = div
meaning And = 0 -- <- your code here (remember an Int is false if it is (==0) and true othe
meaning Or = 0 -- <- your code here

data Exp =
    | Const Int
    | Var String
    | BinExp BinOp Exp Exp
    | Let String Exp Exp
    | If Exp Exp Exp
    | FF
    | Not Exp
      deriving Show

type Assignment = (String -> Int)
update :: (String,Int) -> Assignment -> Assignment
update (x,v) f = (y -> if x == y then v else f y)

a0 :: Assignment
a0 x = error ("undefined variable " ++ x)

eval :: Assignment -> Exp  -> Int
eval a (Const k) = k
eval a (Var s) = a s
eval a (BinExp op e1 e2) = (meaning op) (eval a e1) (eval  a e2)
eval a (Let x e1 e2) = 0 -- <- your code here
eval a (If b e1 e2) = 0 -- <- your code here
eval a FF = 0 -- <- your code here
eval a (Not e) = 0 -- <- your code here
```

For a `Let` expression of the form `Let x e1 e2` evaluate `e1` in using the current assignment - call the resulting value `v`. Now, make a new assignment in which the string `x` gets bound to `v` and use that assignment to evaluate `e2`.

[1]This is like C, C++ and Ruby, 0 is interpreted as false and *any* non-zero value is interpreted as true.

Exercise 0.1. You need to fill in the missing code.

Exercise 0.2. You can check your code ont he test cases included - but ALSO write some test cases of your own. The extensiveness of your test code will count toward the credit on this homework. (Stuff like having a `Let`-expression as the Boolean in an `If-then-else` construct is the kind of creative and interesting test cases we hope to see.)