HW 15 **Prof. Caldwell**
Due: 30 October 2008 **COSC 3015**

Here is some code for the calculator presented in class today with some additional constructors (`if-then-else`, `FF`, `Not`, `And`, and `Or`).

```
data Exp = N Int
         | V String
         | Add Exp Exp
         | Mul Exp Exp
         | Let String Exp Exp
         | If Exp Exp Exp
         | FF
         | Not Exp
         | And Exp Exp
         | Or Exp Exp
  deriving (Eq,Show)
```

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 `N 19` is interpreted as true and `N 0` is interpreted as false.

```
true x = x /= 0

eval m (N k) = k
eval m (V x) = m x
eval m (Add e1 e2) = (eval m e1) + (eval m e2)
eval m (Mul e1 e2) = (eval m e1) * (eval m e2)
eval m (Let x e1 e2) = eval m' e2
  where m' z = if z == x then (eval m e1) else m z
eval m FF = {- your code goes here -}
eval m (And e1 e2) = fromEnum (true (eval m e1) &&  true (eval m e2))
eval m (Or e1 e2) =  {- your code goes here -}
eval m (If e1 e2 e3) = {- your code goes here -}
eval m (Not e1) = {- your code goes here -}
```

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

Exercise 0.2. Write test cases for your code. The extensiveness of your test code will count for more than half 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.)

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