CS 421 --- Interpreter Activity 1

Manager	Keeps team on track	
Recorder	Records decisions	
Reporter	Reports to class	
Reflector	Assesses team performance	

Code

Here is part of the code for the `i3.hs` interpreter.

```
-- The Types

data Val = IntVal Integer
   deriving (Show,Eq)

data Exp = IntExp Integer
         | IntOpExp String Exp Exp
   deriving (Show,Eq)

type Env = [(String,Exp)]

-- Evaluator

intOps = [ ("+",(+))
         , ("-",(-))
         , ("*",(*))
         , ("/",div)]

liftIntOp f (IntVal i1) (IntVal i2) = IntVal (f i1 i2)
liftIntOp f _           _           = 0

eval :: Exp -> Env -> Val
eval (IntExp i) _ = IntVal i

eval (IntOpExp op e1 e2) env =
  let v1 = eval e1 env
      v2 = eval e2 env
      Just f = lookup op intOps
  in liftIntOp f e1 e2
```

Problem 1) There are three types declared for this interpreter. What do each of them do?

Problem 2) The `Val` type only has integers. Why not make eval output integers directly then?

Problem 3) There are two bugs in the above code. What are they?

Problem 4) Add variables to this. To do this you need to add a constructor to `Exp` and a clause to `eval`.

Problem 5) If there's time: add comparison operations to the language. You will need a separate variable `compOps` to do this, another constructor for `Exp`, and another clause for `eval`. You may need another lifting function as well. Why can't you just combine this with `intOps`?

Interpreter Activity 1--- Reflector's Report

Manager	Keeps team on track	
Recorder	Records decisions	
Reporter	Reports to Class	
Reflector	Assesses team performance	

1. What was a strength of your team's performance for this activity?

2. What could you do next time to increase your team's performance?

3. What insights did you have about the activity or your team's interaction today?