Interpreter Activity 2

Mattox Beckman

Code

Here is part of the code for the `i6.hs` interpreter.

```
-- The Types

data Val = IntVal Integer
         | BoolVal Bool
         | ClosureVal String Exp Env
   deriving (Show,Eq)

data Exp = IntExp Integer
         | IntOpExp String Exp Exp
         | CompOpExp String Exp Exp
         | VarExp String
         | IfExp Exp Exp Exp
         | LetExp String Exp Exp
         | FunExp String Exp
         | AppExp Exp Exp
   deriving (Show,Eq)

type Env = [(String,Val)]

-- Evaluator

type IntOpEnv = [(String, Integer -> Integer -> Integer)]

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

compOps = [ ("<",(<))
          , (">",(>))
          , ("==",(==))
          , ("<=",(<=))
          , (">=",(>=))
          , ("/=",(/=)) ]

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

liftCompOp f (IntVal i1) (IntVal i2) = BoolVal (f i1 i2)
```

```haskell
liftCompOp f _           _           = BoolVal False

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

eval (CompOpExp op e1 e2) env =
  let v1 = eval e1 env
      v2 = eval e2 env
      Just f = lookup op compOps
  in liftCompOp f v1 v2

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

eval (VarExp s) env =
  case lookup s env of
     Just v -> v
     Nothing -> error $ "Undefined: " ++ s

eval (IfExp c t e) env =
  case (eval c env) of
    BoolVal True -> eval e env
    _            -> eval t env

eval (LetExp v e1 e2) env =
  let v1 = eval e1 env
   in eval e2 ( (v,v1):env)

eval (FunExp v e1) env =
  ClosureVal v e1 env

eval (AppExp e1 e2) env =
  let ClosureVal v e3 cenv = eval e1 env
      arg = eval e2 cenv
  in eval e3 ((v,arg):env)
```

1. With a partner, code review this. Some bugs have been inserted!

2. Can you modify `let` to be able to define more than one variable at a time? Also, allow for functions to take multiple parameters.

3. Can you add lists to the language? Add `cons`, `nil`, `head`, `tail`, and `empty?` to the language.