PreL.hs

1: {- Author: Richard Eisenberg
2:    File: PreL.hs
3: -}
4:  Defines a main action for the Preλ interpreter, as well as functions
5:    that combine multiple interpretation phases.
6: -}
7: 
8: -- The listing of these modules in the parentheses here make it so that
9: -- all the functions in those modules are available in, e.g., GHCi when
10: -- you load just PreL.hs.
11: module Main (
12:   module Main,
13:   module Eval,
14:   module Parser,
15:   module Lexer,
16:   module Token,
17:   module Syntax
18:   ) where
19: 
20: import Control.Exception
21: import System.Exit
22: import Control.Monad
23: 
24: import Syntax
25: import Parser
26: import Lexer
27: import Eval
28: import Token
29: 
30: main :: IO ()
31: main = do
32: 
33:   -- primary user interaction commands
34:   putStrLn ""
35:   putStrLn "Enter an expression:"
36:   expr_string <- getLine
37: 
38:   -- allow users to quit
39:   when (expr_string == "quit")
40:     exitSuccess
41: 
42:   -- This code runs evalString in a way that, if evalString calls 'error',
43:   -- the program will not immediately abort. The Haskell features used here
44:   -- are beyond the scope of CS245. The curious may enjoy looking these
45:   -- functions up online.
46:   catch (do value <- evaluate (evalString expr_string)
47:           print value)
48:         (\ (SomeException e) -> print e)
49: 
50:   -- And do it again.
51:   main
52: 
53:   -- Lex and parse an expression string.
54:   -- Calls 'error' if the input is somehow malformed.
55:   lexParse :: String -> Expr
56:   lexParse = error "unimplemented"
57: 
58:   -- Lex, parse, and evaluate an expression string.
59:   -- Calls 'error' if the input is somehow malformed or cannot be
60:   -- evaluated. This might conceivably loop if the input expression
61:   -- does not terminate.
62:   evalString :: String -> Value
63:   evalString str
64:     = eval (lexParse str)