INFOB3TC - Extra Question for revision

Lawrence Chonavel

February 2025

1 (15p total) Compiler Checks

Consider the language generated by:

```
Exp
  ::= λ Var. Exp
  |   Var
  |   Exp Exp
  |   (Exp , Exp)
  |   case Exp of (Var , Var) -> Exp
  |   (Exp)
```

And its Haskell implementation:

```haskell
data Exp
  = ELam Var Exp
  | EVar Var
  | EApp Exp Exp
  | EPair Exp Exp
  | ECase Exp Var Var Exp

type Var = String

data EAlg e = EAlg
  { eLam  :: Var-> e -> e
  , eVar  :: Var -> e
  , eApp  :: e -> e -> e
  , ePair :: e -> e -> e
  , eCase :: e -> Var -> Var -> e -> e
  }

foldE :: EAlg e -> Exp -> e
parse :: String -> Exp
```

You are tasked with writing a *linter*, to detect suspicious code.

Your code may call basic Haskell functions without definition, e.g.

```haskell
elem :: (Eq a) => a -> [a] -> Bool
```

Your linter may not assume that all functions are well-scoped or well-typed.

1.1 (15p) Implement `findShadowed :: EAlg ([Var] -> [Var])`

This algebra should return a function, taking a list of all variables in scope, and returning a list of those variables which the expression shadows.

For example, it should satisfy:

```haskell
foldE findShadowed
  (parse "λ a. λ b. b (λ a. a b (λ b. λ b. x))")
  []
== ["a", "b", "b"]
```

```
foldE findShadowed
  (parse "case p of (a , a) -> sqrt a")
  []
== ["a"]

foldE findShadowed
  (parse "λ x. λ f. case f x of (y , x) -> y")
  []
== ["x"]

foldE findShadowed
  (parse "λ f. ((λ x -> f (x x)) (λ x -> f (x x)))")
  []
== []
```

The list should contain one entry per shadowed variable binding, in order.