Madhuri Pyreddy
9/26/19

CSCI 117 Lab 6

Part 1:

```
local Or L Ln Result Resultn in
  Or = fun {$ L}
    case L
    of nil then true
    [] '|'(1:H 2:T) then
      if (H == true) then O in
        O = {Or T}
        O
      else
        false
      end
    end
  end
  L = [true true false]
  Result = {Or L}
  skip Browse Result

  Ln = [true true true]
  Resultn = {Or Ln}
  skip Browse Resultn
end

local NElems Tree Result in
//number of elements in Tree 2
  NElems = fun {$ T}
    case T
    of leaf(1:N) then 1
    [] node(1:Leaf 2:Left 3:Right) then
      (Leaf + ({NElems Left} + {NElems Right}))
    end
  end
    Tree = node(1:3 2:leaf(1:3) 3:leaf(1:1))
    Result = {NElems Tree}
    skip Browse Result
end
```

```
local NumEmpty Tree Result in
NumEmpty = fun {$ T}
case T
of empty then 1
 [] node(1:E 2:TL 3:TR) then L R in
        L = {NumEmpty TL}
        R = {NumEmpty TR}
        (L + R)
     end
  end
   Tree = node(1:5 2:node(1:6 2:empty 3:empty) 3:empty)
    Result = {NumEmpty Tree}
    skip Browse Result
end
```

<u>Part 2:</u>

```
local Y Min Five Seven Result G H L1 L2 F1 F2 G1 G2 D1 D2 Z1 Z2 AB1 AB2 EF1 EF2 in
   Y = true
   Five = 5
   Seven = 7
   Min = proc {$ X Z Out}
       if Y then
          Out = X
       else
          Out = Z
       end
   {Min Five Seven Result}
   skip Browse Result
end
    G = 5
    H = pair(1:G 2:G)
    case H of pair(1:Z 2:W) then
      local T in
        {IntMinus Z W T}
        skip Full
      end
```

```
      else
        skip Basic
    end
    L1 = person(kid:G1 drink:F1 food:D1)
    L2 = person(kid:G2 drink:F2 food:D2)
    Z1 = kid(pet:AB1 hobby:EF1)
    Z2 = kid(pet:AB2 hobby:EF2)
    AB1 = cat()
    D1 = meatloaf()
    F1 = juice()
    EF1 = swimming()
    L1 = L2
  skip Full
end
local X in
local X in
          X = 5
     skip Full
    end
end
```

b. Each time they are displayed, the stores and the environment appear different from each other. For the min procedure with a non-empty closure, it is part of the syntax statement that includes Y Min Five Seven Result as variables and S as a sequence of statements (the body of the procedure). Once such a procedure value is assigned to a variable, then that variable can be used in a procedure call. The procedure call includes other variables X and Z. It assigns Y as a min variable and then starts the procedure for Min Y. It later assigns X and Z as the variables for comparing Y. The reason is that it will determine which variables have the minimum value. If . If it is true, then it will output X. If not, it will output Z. It will determine to see which variable has the minimum value out of the 3 variables declared in the procedure. The environment in min procedure Y do not change at all because the variables Y, Min, Five, Seven, and Result are mapped to reference other integers. The stack doesn't build up in the min procedure Y. The case statement is used for pattern matching with the record list variable. If it matches, it moves on to statement one <s1> and then to statement two <s2>. It continues with the rest of the execution statements in the stack until the variables are not bound with what we are matching. Once that happens, it should give an error. It will bind the unbound variable G to the integer 1, and add information to the store. For H, however, it will bind the unbound pair of Z and W to the integers 1 and 2. Then comes T, which is the sequence . In sequence, it will bind to Z, W, and T and skips. It ends the program. The environment for case statement has variables G, H, Z, W, and T mapped to reference integers and the stack for case statement builds up because it includes variables that are paired to other variables like "((11), pair(1:10 2:10))", "((10), 5)", and "((6), proc)". In unification of nested records, it will assign L1, L2, F1, F2, G1, G2, D1, D2, Z1, Z2, AB1, AB2, EF1, and EF2. It unifies these variables and sets it equal to other variables like kid, drink, food, pet, and hobby . It assigns integers for those variables. The environment for unification of nested records includes L1, L2, F1, F2, G1, G2, D1, D2, Z1, Z2, AB1, AB2, EF1, and EF2 mapped to other integers. The stack builds up, including variables paired with other variables like "((18, 19), meatloaf())", "((14, 15), juice())", and "((12, 13), person(drink:14 food:18 kid:16))".

Extra Credit:

```
local NumNodes Tree Out  Result in
NumNodes = fun {$ T}
case T
of empty then 0
 [] node(1:E 2:TL 3:TR) then L R in
        L= {NumNodes TL}
        R = {NumNodes TR}
       (L + R)
 end
end
Tree = node(1:5 2:node(1:6 2:empty 3:empty) 3:empty)
    Result = {NumNodes Tree}
    skip Browse Result
  end
local SumT2It Tree Result in
  SumT2It = fun {$ T}
    SumH = fun {$ TList S}
      case TList
      of nil then S
      [] '|'(1:H 2:T) then case H
        of empty then O in
          O = {SumH T S}
          O
        [] node(1:E 2:TL 3:TR) then O in
          O = {SumH (TL|(TR|T)) (S+E)}
          O
        end
```

```
      end
    end Out in
    Out = {SumH (T|nil) 0}
    Out
  end
Tree = node(1:5 2:node(1:6 2:empty 3:empty) 3:empty)
  Result = {SumT2It Tree}
  skip Browse Result
end
```