92

Andrew Giardina 4/9/2021

Language Study

Homework 6

Problem 1

```
substitute(X,Y,[ ],[ ]).                        // Base Case: Empty list, done

substitute(X,Y,[X|L1],L2) ←                     // X exists, replace with Y, recur
      substitute(X,Y,L1,[Y|L2]).

substitute(X,Y,[N|L1],[N|Ys]) ←                 // N is head, N ≠ X, no replace, recur
      X ≠ N,
      substitute(X,Y,L1,L2).
```

This will never get to the base case

-2

Problem 2

```
no_doubles([ ],[ ]).           // Base Case: Empty Lists

no_doubles([H|L1],L2) ←        // H also exists in L1, Remove all 'H's, recur with NewL
      member(H,L1),
      ~~delete(L1,H,NewL),~~
      no_doubles(L1,~~NewL~~ L2).

no_doubles([H|L1],[A|L2]) ←   // H not dupped in L1, recur to next element
      no_doubles(L1,L2).
```

-3

Problem 3

```
sum_tree(void,0).                          // Base Case: Sum of empty tree is 0

sum_tree(tree(Root,Left,Right),Sum) ←      // Rec Case: Tree not empty
      sum_tree(Left,LSum),                 // Recur on Left Tree
      sum_tree(Right,RSum),                // Recur on Right Tree
      plus(LSum,Root,TSum),                // Now add Left Sum to Root, = Temp Sum
      plus(RSum,TSum,Sum).                 // Add Right Sum to Temp Sum, = Sum
```

Problem 4

```
path(X,void,[]).                           // Base Case: No path for empty tree

path(X,tree(X,Left,Right),[~~Path~~|X]).   // Base Case: X found, add to existing Path

path(X,tree(Y,Left,Right),Path) ←          // Rec Case: X !found
      X ≠ Y,                               // Confirm X /= Y
      append(Path,Y,NewPath),  ]           // Append Y to the back of current path
      path(X,Left,NewPath),    ] ?         // Recur to Left with NewPath
      path(X,Right,NewPath).   ]           // Recur to Right with NewPath
                                 -3
```