Patrick Austin  
CS 326

Homework # 7

11/29/2015

% Patrick Austin

% CS 326 Homework 7

% 11/29/2015

% 1.

% Reverse of an empty list is an empty list.

reverse([], []).

% Reverse of a list with only a head is a list consisting of the head.

reverse([H|[]], [H]).

% Reverse of a list with head and non-empty tail is append of reverse of tail with head.

reverse([H|T], X) :- reverse(T, Y), append(Y, [H], X).

% 2.

% Take with 0 elements requested is an empty list.

take(\_, 0, []).

% Take of an empty list for any number of elements requested is an empty list.

take([], \_, []).

% Take of an extant list for non-zero N is a list with existing head and tail of take with N-1 elements requested.

take([H|L], N, [H|L1]) :- N > 0, X is N-1, take(L,X,L1).

% 3.

% Definitions for a tree, as specified.

tree(nil).

tree(node(\_,Left,Right)) :- tree(Left), tree(Right).

% 3a.

% nleaves of an empty tree is 0.

nleaves(nil, 0).

% nleaves of a tree with two nil children is 1. Have reached the bottom, can cut.

nleaves(node(\_,nil,nil), 1) :- !.

% nleaves of a tree with child nodes is nleaves of left subtree plus nleaves of right subtree.

nleaves(node(\_,Left,Right), N) :- nleaves(Left,LeftN), nleaves(Right,RightN), N is LeftN+RightN.

% 3b.

% treeMember of a node containing the target value is true. Have found the value, can cut.

treeMember(X, node(X,\_,\_)) :- !.

% treeMember of a node not containing the target value is treeMember of left and right subtrees.

treeMember(X, node(\_,Left,Right)) :- treeMember(X,Left); treeMember(X,Right).

% 3c.

% preOrder of an empty tree is the empty set.

preOrder(nil, []).

% preOrder of a tree with child nodes is append of:

% a list with this node’s value for the head and the preorder of the left subtree for the tail,

% and preorder of the right subtree

preOrder(node(X,Left,Right), L) :- preOrder(Left,LeftL),

preOrder(Right,RightL), append([X|LeftL], RightL, L).

% 3d.

% height of an empty tree is 0.

height(nil, 0).

% height of a node with children is the larger of the heights of the subtrees plus 1, for this level.

height(node(\_,Left,Right), H) :- height(Left,H1), height(Right,H2), H is max(H1,H2)+1.

% 4.

% insert into an empty list is a list consisting of the inserted value.

insert(X, [], [X]).

% insert into a list where target value would go after the first element:

% a list with original head and tail of insert of target value into tail.

insert(X, [H|Y], [H|Z]) :- X>H, insert(X,Y,Z).

% insert into a list where the target value would go first:

% a list with target value at the head and the previous head and tail in the new tail.

insert(X, [H|Z], [X|[H|Z]]) :- X=<H.

% 5.

% flatten of an empty list is an empty list.

flatten([], []).

% flatten of a list with an empty list for the head is flatten of just the tail.

flatten([[]|A], B) :- flatten(A,B).

% flatten of a list where the head is atomic is a list with that head and a flattened tail.

flatten([A|X], [A|Y]) :- atomic(A), flatten(X,Y).

% flatten of a list where the head is non-atomic is append of flatten on head and flatten on tail.

flatten([A|T], B) :- flatten(A,X), flatten(T,Y), append(X,Y,B).