0: Lists without Lisp!

(a) Prove: Concat is symmetric across []. That is, prove for all lists L, $concat([], L) = concat(L, [])$. We do this by induction.
Base Case: $L = []$. $concat([], L) = concat([], []) = [] = concat([], []) = concat(L, [])$.
Induction Hypothesis: $concat(L, []) = concat([], L)$ for some list L.
Induction Step:

$$\begin{aligned} concat(x :: L, []) &= x :: concat(L, []) && \text{Definition of concat} \\ &= x :: concat([], L) && \text{By I.H} \\ &= concat(x :: [], L) && \text{Definition of concat} \end{aligned}$$

Thus, the inductive step holds. Since both the base case and inductive step hold, we have shown that for all lists L, concat is symmetric across [].

(b) Prove: For all lists A, B, C, $concat$ is associative. That is: $concat(concat(A, B), C) = concat(A, concat(B, C))$. We do this by Induction.
Base Case: $A = []$.

$$\begin{aligned} concat(concat(A, B), C) &= concat(concat([], B), C) && \text{A = []} \\ &= concat(B, C) && \text{Definition of concat} \\ &= concat([], concat(B, C)) && \text{Definition of concat} \\ &= concat(A, concat(B, C)) && \text{A = []} \end{aligned}$$

Induction Hypothesis: $concat(concat(A, B), C) = concat(A, concat(B, C))$ for some lists A, B, C.
Induction Step:

$$\begin{aligned} concat(concat(x :: A, B), C) &= concat(x :: concat(A, B), C) && \text{Definition of concat} \\ &= x :: concat(concat(A, B), C) && \text{Definition of concat} \\ &= x :: concat(A, concat(B, C)) && \text{By I.H.} \\ &= concat(x :: A, concat(B, C)) && \text{Definition of concat} \end{aligned}$$

Thus, the inductive step holds. Since both the base case and inductive step hold, we have shown that for all lists A, B, C, $concat$ is associative.

(c) Prove: $rev(concat(A, B)) = concat(rev(B), rev(A))$ for all lists A and B. We do this by structural induction.

Case($A = []$):

$$\begin{aligned}
rev(concat(A, B)) &= rev(concat([], B)) && \text{A = []}\\
&= rev(B) && \text{Definition of concat}\\
&= concat([], rev(B)) && \text{Definition of concat}\\
&= concat(rev(B), []) && \text{concat is symmetric (part a)}\\
&= concat(rev(B), rev([])) && \text{Definition of rev}\\
&= concat(rev(B), rev(A)) && \text{A = []}
\end{aligned}$$

Induction Hypothesis: $rev(concat(A, B)) = concat(rev(B), rev(A))$ for some lists A and B.
Case($A = x :: A$):

$$\begin{aligned}
rev(concat(x :: A, B)) &= rev(x :: concat(A, B)) && \text{Definition of concat}\\
&= concat(rev(concat(A, B)), x :: []) && \text{Definition of rev}\\
&= concat(concat(rev(B), rev(A)), x :: []) && \text{By I.H.}\\
&= concat(rev(B), concat(rev(A), x :: [])) && \text{concat is associative (part (b))}\\
&= concat(rev(B), rev(x :: A)) && \text{Definition of rev}
\end{aligned}$$

Thus, since both the base case and inductive step hold for all lists A and B, we have shown by structural induction on A that $rev(concat(A, B)) = concat(rev(B), rev(A))$.

1: Proving BST Insertion Works!

(a) Prove: For all $b \in \mathbb{Z}$, $v \in \mathbb{Z}$ and all trees T, if $less(b, T)$, and $b > v$, then $less(b, insert(v, T))$. We do this by Structural Induction.

Base Case ($T = Nil$): Show that if $less(b, T)$ and $b > v$, then $less(b, insert(v, Nil))$.
If T is Nil, then $less(b, Tree(v, Nil, Nil))$ depends on $v < b$, $less(b, Nil)$, and $less(b, Nil)$ by the BST definition. It is given that $b > v$ and we know that $less(b, Nil)$ is true by the BST definition. Therefore the base case holds.

Induction Hypothesis: Let x be an arbitrary integer. We can define Tree $T = Tree(x, L, R)$. If $less(b, T)$ and $b > v$, then $less(b, insert(v, T))$ holds for the left subtree (L) and the right subtree (R).

Induction Step: If we let $less(b, T)$ be true, it follows that $v < b$ and $less(b, L)$ and $less(b, R)$. To prove the beginning statement, we must show that $less(b, insert(v, T))$ is true. There are two cases to consider: 1. $v < b$ and 2. $v \geq b$.
Case 1:

$$
\begin{aligned}
less(b, insert(v, T)) &= less(b, insert(v, Tree(x, L, R))) && \text{Definition of BST} \\
&= less(b, Tree(x, insert(v, L), R)) && \text{Definition of BST insert} \\
&= x < b \text{ and } less(b, insert(v, L)) \text{ and } less(b, R) && \text{Definition of BST less}
\end{aligned}
$$

It is given that $x < b$ and $less(b, R)$ is true. Therefore, the above becomes:

$$less(b, insert(v, T)) = less(b, insert(v, L)) = \text{true by I.H.}$$

Case 2:

$$
\begin{aligned}
less(b, insert(v, T)) &= less(b, insert(v, Tree(x, L, R))) && \text{Definition of BST} \\
&= less(b, Tree(x, L, insert(v, R))) && \text{Definition of BST insert} \\
&= x < b \text{ and } less(b, L) \text{ and } less(b, insert(v, R)) && \text{Definition of BST less}
\end{aligned}
$$

It is given that $x < b$ and $less(b, L)$ is true. Therefore, the above becomes:

$$less(b, insert(v, T)) = less(b, insert(v, R)) = \text{true by I.H.}$$

Since both cases hold and the base case hold, we have shown that for all $b \in \mathbb{Z}$, $v \in \mathbb{Z}$ and all trees T, if $less(b, T)$, and $b > v$, then $less(b, insert(v, T))$.

(b) Prove: For all trees T and all $v \in \mathbb{Z}$, if $isBST(T)$, then $isBST(insert(v,T))$. We do this by structural induction.

Base Case ($T = Nil$):
$isBST(insert(v,T)) = isBST(insert(v,Nil)) = isBST(Tree(v,Nil,Nil)) = less(x,Nil)$ and $isBST(Nil)$ and $greater(x,Nil)$ and $isBST(Nil)$ by the definition of isBST. By the definitons of less and greater, the $less(x,L)$ and $greater(x,R)$ terms both evaluate to true. Additionally, $isBST(Nil)$ also evaluates to true by the definition of isBST. Therefore, the base case holds.

Induction Hypothesis: Let x be an arbitrary integer. We can define Tree $T = Tree(x,L,R)$. If $isBST(T)$, then $isBST(insert(v,T))$ holds for the left subtree (L) and the right subtree (R).

Induction Step: If we let $isBST(T)$ be true, it follows that $less(x,L)$ and $isBST(L)$ and $greater(x,R)$ and $isBST(R)$ by the definition of isBST. To prove the beginning statement, we must show that $isBST(insert(v,T))$ is true. There are two cases to consider: 1. $v < b$ and 2. $v \geq b$.

Case 1:

$$isBST(insert(v,T)) = isBST(Tree(x,insert(v,L),R)) \qquad \text{Definition of BST insert}$$

We know that $isBST(Tree(x,insert(v,L),R)) = less(x,insert(v,L))$ and $isBST(insert(v,L))$ and $greater(x,R)$ and $isBST(R)$ by the definition of isBST. It is given that $greater(x,R)$ and $isBST(R)$ both evaluate to true. Additionally, since it is given that $isBST(L)$ is true, then $isBST(insert(v,L))$ also evaluates to true by the I.H. Therefore, the final equation evaluates to:

$$isBST(insert(v,T)) = less(x,insert(v,L)) = \text{true by part (a).}$$

Case 2:

$$isBST(insert(v,T)) = isBST(Tree(x,L,insert(v,R))) \qquad \text{Definition of BST insert}$$

We know that $isBST(Tree(x,L,insert(v,R))) = less(x,L)$ and $isBST(L)$ and $greater(x,insert(v,R))$ and $isBST(insert(v,R))$ by the definition of isBST. It is given that $less(x,L)$ and $isBST(L)$ both evaluate to true. Additionally, since it is given that $isBST(R)$ is true, then $isBST(insert(v,R))$ also evaluates to true by the I.H. Therefore, the final equation evaluates to:

$$isBST(insert(v,T)) = greater(x,insert(v,R)) = \text{true by part (a).}$$

Since both cases hold and the base case hold, we have shown that for all trees T and all $v \in \mathbb{Z}$, if $isBST(T)$, then $isBST(insert(v,T))$.