Invariant for BitList:

BitList l is a well formed BitList if every element in the list is either Zero or One with a balanced binary tree with depth equal to the index within the list of the tree

For cons\_helper: We will rename the parameters x’ and l’ to distininguish them from the parameters of cons

Prove: that cons\_helper x’ l’ return s a well formed list

Invariant of cons\_helper: x’ is an balanced binary tree with depth equal to length of list l minus length of list l’

Base case: l’ is the empty list and x’ is a leaf (the BitList invariant holds for l’ because it is empty)

Cons\_helper with return [One(x’)] which is a well formed list because the zero-th element (the only element in the list) has depth 0.

Inductive Hypothesis: l’ is a well formed bit list with length n and x’ is a leaf, then cons\_helper x’ l’ will return a well formed bit list

Inductive Step: l’ is a well formed bit list with length n+1 and x’ is a leaf

Case 1:

l’ is the same as (l1’ @ Zero) where l1’ is a well formed list with length n

cons\_helper x’ l1’ returns a well formed BitList.

If cons\_helper x’ l1’ returns a list of length n then cons\_helper x’ l’ will return (cons\_helper x’ l1’ @ Zero) which is a well formed list

If cons\_helper x’ l1’ returns a list of length n+1 then cons\_helper x’ l’ will return (l2’ @ One(x’’)) where l2’ is a list of length n with all zeros and x’’ is a tree of depth n+1, which means cons\_helper x’ l’ is a well formed bit\_list

Case 2:

l’ is the same as (l1’ @ One (x2’)) where l1’ is a well formed list with length n and x2’ is a balanced binary tree with depth n+1

cons\_helper x’ l1’ returns a well formed BitList.

If cons\_helper x’ l1’ returns a list of length n then cons\_helper x’ l’ will return (cons\_helper x’ l1’ @ One(x2’)) which is a well formed BitList because cons\_helper returned a well formed BitList with length n and x2’ is a balanced binary tree with depth n+1

If the cons\_helper x’ l1’ returns a list of length n+1 then cons\_helper x’ l’ will return (l2’ @ Zero @ One(Node(x1’, x2’))). This is a well formed BitList because l2’ @ Zero is a BitList of zeros with length n+1 and x1’ and x2’ are tree with depth x+1 meaning that Node(x1’, x2’) has depth n+2. Therefore, the returned list is well formed by the definition of a well formed BitList.

I have shown that given a leaf x’ and a well formed list l’ cons\_helper x’ l’ will return a well formed list

Cons takes in x and l. x is an element with the type of list l, and l is a well formed BitList. Cons returns cons\_helper Leaf(x) l. cons\_helper Leaf(x) l returns a well formed BitList by the above proof.