4.1

-2 for domain

Domain: ~~all values of type order~~ for values t of type tree and m of type int

Property P(t): If max t = Some m, then Member (m,t)

Order: R(t1,t2): t1 is a proper substructure of t2

Partition: {Leaf} {Branch (l,k,r) | l and r are values of type int tree and k is a value of type int}.

Case 1: t = Leaf

Property holds vacosly because max {Leaf} = None

Case 2: t = Branch (l,k,r)

By assumption, we know max t = Some m no!

2nd Induction

Domain: ~~All values of type order~~ for int values m and k

Property Q(o): if compare_int m k = o, then P(t)

Order: R(o1,o2) false (empty relation)

Partition: {Less}, {Equal}, {Greater}

Case a: o = Less

This situation is not delt with in max, so holds

Case b: o = Equal

Less Greater and Equal are not in the max function. You should have broken up the function into None and Some m, -6

If o = Equal, by definition of max

max k = Some m, by P(t)

Member(m,k), and because k is a proper substructure of t, Member(m,t)

-5 inductive case needs work

Case c: o = Greater

If o = greater, by definition of max

max r = Some m, by P(t)

Member(m,r), because r is a proper substructure of t

Member(m,t)

this is the right idea, but your partitioning in this problem is wrong

∴ Because either o = Equal or o = Greater will be true P(t).

17/30

4.2

Domain: ~~All values of type order~~ -2 domain

Property P(t): If max t = Some m, then AllLess(m, delete m t) and If max t = None, then t = Leaf, for values of t, t is type tree and m, type int

Order: R(t1, t2): t1 is a proper substructure of t 2.

Partition: {Some m | m is a value of type int} {None}

your partitions should be leaf and branch

Case 1: max t = None.

-4 no leaf base case

by the definition of max, max t returns None when t is matched with Leaf, therefore t = Leaf.

-18 many issues with the inductive case

You can't use intuition like m is the greatest int value without proving it.

Case 2: max t = Some m

- by the definition of max on BST(t), m is the greatest int value in the tree t.
- All Less (m, delete m t) is true if m is greater than every value in the tree delete m t.
- We know m is the greatest value in t, so if delete correctly removes m from t, P(t) holds.

This is intuition, and not rigorous.

- By lemma 1, If max t = Some m, AllLess(m, delete m t)

Lemma 1: for values i of type int and tr of type tree, if Member(i, tr) delete i tr removes i from tr.

Domain: ~~all values of type order~~

Property P(i): If Member(i, tr), delete i tr removes i from tr

Order: R(t1, t2): t1 is a proper substructure of t2

Partition: {Member(i, tr) = true} {Member(i, tr) = false)}

Case 1: Member(i, tr) = false

by thm 4.1 this is an impossible case.

Case 2: Member(i, tr) = true

~ by def of delete with the understanding that Member(i, tr) delete will reach the 'Equal' case in the 2nd matching

~ From here there are 2 cases the right side of the tree matches with Leaf or —.

~If r matches with Leaf, this means there is nothing in the right side, so the only remaining part is the left side, so delete returns the left side of br.

~If r matches with _, delete will then check if l is a leaf or not, it matches max l with Some m and None

- ☆ in the case of None, this means l is empty and delete will return r.
- ☆ in the case of Some m, this means l is not empty, so delete returns a tree with both r and l.

6/30