4.1

Domain: all values of type order
Property P(t): If max t = Some m, then Member (m, t) for value t of type tree and m of type int
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 vacously because max {Leaf} = None

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

By assumption, we know max t = Some m

2nd Induction

Domain: All values of type order
Property Q(o): if compare_int m k = 0, then P(t) for int values m and k
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

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)

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)

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

4.2

Domain: all values of type order
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: tree and m, type int
order: R(t1, t2) : t1 is a proper substructure of t2.
Partition: {Some m | m is a value of type int} {None}
Case 1: max t = None.
by the definition of max, max t returns None
when t is matched with leaf., therefore
t = Leaf.
Case 2: max t = Some m
- by the definition of max on BST(t), m is the greatest int value in the tree t.
- AllLess(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 it delete correctly removes m from t, P(t) holds.
- 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 mirrores 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 tr.

~ 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.