Marcus Bischof - 260503230

Q1 Assignment 2

Comp 302

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Proof by structural induction on t

Base Case : reflect ( reflect LF ) = LF

By definition this hold : reflect ( reflect LF ) == reflect LF == LF

BOTH induction hypotheses :

reflect (reflect t1 ) == t1 && reflect (reflect t2) == t2

Need to show that -- reflect (reflect ( Br (x, t1, t2))) == Br (x, t1, t2)

^^^ *This proves that if you reflect on a node twice, you get that node in the tree*

Simplifying we get ->

= reflect (reflect ( Br (x, t1, t2)))

= reflect ( Br (x, reflect t1, reflect t2)))

= Br (x, reflect ( reflect t1 ), reflect ( reflect t2 ))))

ih1 = Br (x, t1, reflect ( reflect t2 ))))

ih2 = Br (x, t1, t2 )

**DONE**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Prove that size m = size'(m,0)

ih1 --> size L = size'(L,0)

ih2 --> size R = size'(R,0)

lemma: size m + acc == size' (m, acc)

Induction on m

Base case m = Empty

--> size Empty + acc == 0 + acc == acc

--> size' (Empty, acc) = acc

^^ *thus two sides are both equal*

Step Case : m = Node (x, L, R)

ih1 --> size L + acc\_L = size' (L, acc\_L)

ih2 --> size R + acc\_R = size' (R, acc\_R)

--> size' (Node(x,L,R),acc) => size'(L,size'(R,x\_acc))

ih2 --> x\_acc = acc\_R => size'(L, size R + (x + acc))

ih1 --> acc = size R + (x + acc) => size L + (size R + (x + acc) )

now use associativity and commutative property to rewrite

=> x + size L + size R + acc

=> size(Node(x,L,R)) + acc

=> x + size L + size R + acc (same as two above!!)