# Design description of the CA208 Assignment

#### Implementing Prolog predicates for non-self-balancing 2-3 Trees

#### by Thomas Hazekamp

The are 4 predicates for this program which all relate to a non-self-balancing 2-3 Tree. The first predicate adds a certain new element to the given tree, if there is only one parent and the new element which is wanting to be added is greater than that parent node, then this section turns into a 3 node from a 2 node. Using recursion, it will find the correct place to be added, this occurs once it finds a leaf node (‘nil’). The add predicate will compare with the left, middle (if exists) and right child to attain the correct path for its final position on the tree.

The second predicate is called member and checks if the given element is part of the given tree. It does this by having a base case if the Node is same as the parent node, this means it is a member of the tree. Using recursion, it will keep going through the nodes (children) and compare if the node is equal to the original given node.

The third predicate is called height which checks the height of the given tree. The base case is an empty tree and is using recursion by adding 1 until it finds the nil leaf, then will get the largest of the heights of all sub trees and return the correct tree height.

Finally, the fourth predicate is called prettyPrint and will print to the console the given tree in a readable way. The base case is when it reaches a leaf (‘nil’) and for the predicate it keeps a number to simulate the depth it is currently at. This position/depth will help print a certain number of spaced text so the node will be in the correct position on screen. This predicate uses an inorder traversal to print all the nodes, first will print the left side of the tree, then the root, followed by the middle and right side respectively.

## Tests and expected results:

##### TESTING FOR ADD

?- add(3, tree(5, nil, nil), T).

T = tree(5, tree(3, nil, nil), nil)

?- add(8, tree(5, nil, nil), T).

T = tree(5, 8, nil, nil, nil).

?- add(33, tree(26, 30, tree(15, nil, nil), tree(27, 29, nil, tree(28, nil, nil), nil), tree(31, 44, nil, tree(33, 40, nil, tree(35, 37, nil, nil, nil), nil), nil)), T).

T = tree(26, 30, tree(15, nil, nil), tree(27, 29, nil, tree(28, nil, nil), nil), tree(31, 44, nil, tree(33, 40, tree(33, nil, nil), tree(35, 37, nil, nil, nil), nil), nil)

?- add(99, tree(19, 35, tree(8, 13, nil, nil, nil), tree(21, nil, nil), tree(46, 50, tree(40, nil, nil), tree(47, 49, nil, tree(48, nil, nil), nil), tree(60, nil, tree(72, nil, nil)))), T).

T = tree(19, 35, tree(8, 13, nil, nil, nil), tree(21, nil, nil), tree(46, 50, tree(40, nil, nil), tree(47, 49, nil, tree(48, nil, nil), nil), tree(60, 99, nil, nil, tree(72, nil, nil))))

##### TESTING FOR MEMBER

?- member(11, tree(5, 10, tree(3, nil, nil), tree(8, 9, tree(3, tree(4, nil, nil), nil), tree(0, nil, nil), tree(1, nil, nil)), tree(12, 20, tree(11, nil, nil), nil, nil))).

true

?- member(72, tree(19, 35, tree(8, 13, nil, nil, nil), tree(21, nil, nil), tree(46, 50, tree(40, nil, nil), tree(47, 49, nil, tree(48, nil, nil), nil), tree(60, nil, tree(72, nil, nil))))).

true

?- member(28, tree(26, 30, tree(15, nil, nil), tree(27, 29, nil, tree(28, nil, nil), nil), tree(31, 44, nil, tree(33, 40, nil, tree(35, 37, nil, nil, nil), nil), nil))).

true

?- member(24, tree(5, 10, tree(3, nil, nil), tree(8, 9, tree(3, tree(4, nil, nil), nil), tree(0, nil, nil), tree(1, nil, nil)), tree(12, 20, tree(11, nil, nil), nil, nil))).

false.

##### TESTING FOR HEIGHT

?- height(tree(5, 10, tree(3, 4, tree(2, tree(1, nil, nil), nil), nil, nil), tree(8, 9, tree(7, nil, nil), nil, nil), nil), H).

H = 4.

?- height(tree(5, 10, tree(3, nil, nil), tree(8, 9, tree(3, tree(4, nil, nil), nil), tree(0, nil, nil), tree(1, nil, nil)), tree(12, 20, tree(11, nil, nil), nil, nil)), H).

H = 4

?- height(tree(19, 35, tree(8, 13, nil, nil, nil), tree(21, nil, nil), tree(46, 50, tree(40, nil, nil), tree(47, 49, nil, tree(48, nil, nil), nil), tree(60, nil, tree(72, nil, tree(75, tree(73, nil, nil), nil))))), H).

H = 6

##### TESTING FOR PRETTYPRINT

?- prettyPrint(tree(5, 10, tree(3, nil, nil), tree(8, 9, tree(3, tree(4, nil, nil), nil), tree(0, nil, nil), tree(1, nil, nil)), tree(12, 20, tree(11, nil, nil), nil, nil))).

?- prettyPrint(tree(3, tree(2, nil, nil), tree(5, tree(4, nil, nil), tree(8, nil, nil)))).

?- prettyPrint(tree(19, 35, tree(8, 13, nil, nil, nil), tree(21, nil, nil), tree(46, 50, tree(40, nil, nil), tree(47, 49, nil, tree(48, nil, nil), nil), tree(60, nil, tree(72, nil, nil))))).

?- prettyPrint(tree(26, 30, tree(15, nil, nil), tree(27, 29, nil, tree(28, nil, nil), nil), tree(31, 44, nil, tree(33, 40, nil, tree(35, 37, nil, nil, nil), nil), nil))).