**CA208 Assignment 2: AVL Trees**

Student Name: David Weir

Student Number: 19433086

**STATEMENT OF NON-PLAGIARISM**

I hereby declare that all information in this assignment has been obtained and presented in accordance with academic rules and ethical conduct and the work I am submitting in this document, except where I have indicated, is my own work.

Signed:

David Weir

**Application Design:**

**bt and empty:**

The design begins by defining empty which is simply an empty with no nodes.

bt defines a non-empty tree, it returns true if the root node given (N) is the root of a binary tree with a left and right subtree, where all the items in the left subtree are less than or equal to the root node and all the items in the right subtree are greater than N. bt is defined in steps. If the subtrees are empty and the root node is not nil or empty, then return true. If we are only given a left subtree (i.e. right is empty), then we check N is not empty and compare each node to ensure they are all less than or equal to the root node N and run using recursions. The same comparisons are performed recursively if only the right tree is non-empty except, we check each node is greater than N. If both the subtrees are populated (i.e. neither are empty) Then we run both comparisons mentioned previously and run bt recursively on the left and right nodes of the previous parent node.

**insert**:

“is true if T2 is the AVL tree resulting from I being inserted into an AVL tree T1.”

Insert starts with a base case where we are inserting a number into an empty list, this predicate returns true if that number is the root node in T2.

Otherwise insert finds the correct position of the new node by comparing it to the root node to find the correct subtree and then recursively calling insert to compare the new node to pre-existing nodes in order to find the correct position.

Bug: If the tree T1 becomes imbalanced (i.e. not AVL) after calling insert then insert cannot balance this tree and it will remain imbalanced.

**display:**

“always true prints the AVL tree to the display.”

Display takes a tree and first prints the root node. It then calls display on the left and right subtrees of the given tree, printing the parent node each time. Display traverses the tree using pre-order traversal and will write each node to the display terminal delaminated by a single whitespace.