In a special binary tree called the *SumTree*, the value of any node in the tree is equal to the sum of all the nodes in the left and right subtrees. If a node has only one child, then that child has to equal the value of the parent node. Leaf nodes in this case are automatically considered a *SumTree*. In this assignment, three functions are used known as: *isSumTree*, *_isSumTree*, and *_Sum*.

The function, *isSumTree*, is basically used as a function to call another private function, *_isSumTree*. In the base case of *isSumTree*, the root is checked, and if it is not empty, then the private recursive function, *_isSumTree*, is called.

_isSumTree is a private function that checks if any inputted tree is a SumTree by adding the left and right subtrees and comparing them to the root node. The base case checks if the current node or any of the left or right child nodes are empty, and returns true if it's the case. Then the function stores the sum of the left subtree in the variable, *left*, and the right subtree in the variable, *right*. Finally, an if-statement containing a recursive call is used to compare the sum of the left and right subtrees to the root node, returning 1 if it's the case. The functions last resort is to return a value of 0 if the tree is not a *SumTree*.

The left and right subtrees are summed using the *_sum* function. The base case for this function returns 0 if the node is empty. Otherwise, a recursive call adds the current node to the left and right child nodes. The result of this function is overall sum of the subtree.