***Beschreibung der Rekursiven Funktionen***

**INSERT**

Die „Insert“ Funktion ist einer der einfachen Funktionen dieser Anwendung. Ein Baum wird immer mit einer Wurzel initialisiert welche den wert „null“ zugewiesen bekommt. Der erste Schritt beim „Insert“ ist es zu prüfen ob der erste Knoten also die Wurzel den Wert „null“ hat somit noch kein Wert in den Baum eingefügt wurde. Sollte dies der Fall sein wird der einzufügende Wert als neuer Knoten in die Wurzel geschrieben. Sollte der Wert des aktuell betrachtenden Knotens nicht „null“ sein und der einzufügende Wert kleiner dem Wert des aktuellen Knotens sein, so wandern wir zu dessen nächsten **linken** Knoten (soweit vorhanden). Sollte der einzufügende Wert größer dem aktuell betrachtenden Knoten sein so wandern wir zu dessen nächsten **rechten** Knoten (soweit vorhanden). Dies wird solange fortgesetzt bis wir entweder an der rechten oder linken Seite eines Knoten (je nach dem einzufügenden Wert) keinen Knoten vorfinden, nun können wir an dieser Stelle einen neuen Knoten mit dem neuen Wert einfügen.

**MINKEY, MAXKEY**

Diese Funktion ist dafür verantwortlich den kleinsten Wert des Baumes zu finden. Da wir wissen, dass ein Binärer-Baum den kleinsten Wert im am weitest linken Knoten speichert müssen wir nur solange nach links den Baum entlang folgen bis wir an den letzten linken Knoten landen um an den kleinsten Wert zu finden. Dies tun wir wie folgt. Wir starten an der Wurzel des Baumes und prüfen ob sich links von diesem ein weitere Knoten befindet, sollte dies der Fall sein wird dieser Vorgang mit diesem Konten wieder holt bis wir auf einen Konten treffen welcher links von sich keinen Konten mehr hat, dann wissen wir, wir haben den kleinsten Wert gefunden. Dieses Prozedere können wir natürlich auch mit der rechten Seite des Baumes machen um den größten Wert des Baumes zu bekommen.

**TOTALKEYNUM**

Diese Funktion liefert uns die Summe aller Werte eines Baumes. Auch hier starten wir wieder an der Wurzel des Baumes und nehmen dessen Wert als ersten zu unserer Variablen „sum“ hinzu. Nun überprüfen wir ob es an der linken oder rechten Seite des Knotens einen weiteren Knoten gibt und führen das Prozedere an diesen Knoten fort und addieren deren rückgabewert zu unsere Summe hinzu. Dies tun wir bis alle Knoten besucht wurden und deren Werte zur gesamten Summe hinzuaddiert wurden.

**TOTOALNODENUM**

Diese Funktion liefert uns die Anzahl aller Knoten zurück. Diese Funktion bedient sich exakt dem gleichen Schema wie die Funktion „TOTALKEYNUM“ mit dem einzigen unterschied, dass nun die Knoten der Funktion TOTALNODENUM übergeben werden. Bei jedem so besuchten Knoten wird die Zahl 1 zu unseren Gesamtanzahl an Knoten hinzuaddiert.

**CHECKAVL**

Diese Funktion ist dafür verantwortlich zu prüfen ob es sich bei einem Binären-Baum um einen ausbalancierten AVL-Baum handelt. Ein AVL-Baum ist ein Baum bei welchem jeder Knoten eine maximale Differenz zwischen der Höhe des linke Teilbaumes und der höhe des rechten Teilbaumes von 1 aufweist. Diese Funktion durchwandert nun auf gewohnte weise einen Binären-Baum von der Wurzel beginnend zu jedem Knoten und wendet auf diesen wieder die Funktion CHEKCAVL an um so jeden einzelnen Knoten nach der oben genannten Bedingung mittels der Funktion BALANCE zu prüfen.

**BALANCE**

Diese Funktion überprüft den übergebenen Knoten auf das AVL-Kriterium. Zu beginn wird eine Variable „diff“ mit dem Wert 0 initialisiert. Diese Variable spiegelt die Balance des teilknoten wider welcher wie schon oben besprochen maximal 1 sein darf. Nun wird geprüft ob es einen Knoten an der **rechten als auch linken**, **nur an der rechten** oder **nur an der linken** Seite des aktuellen Knotens gibt. Auf die so entsprechend gefundenen Knoten wird die Funktion MAXDEPTH angewandt welche die Höhe der jeweiligen Knoten zurückliefert. Jene Seiten, welche **kein** Knoten besitzen erhalten eine Höhe von 0. So erhält man die Differenz beider Knoten. Sollte die Differenz eines einzigen Knotens eines Baumes größer 1 sein, so ist der gesamte Baum KEIN AVL-Baum.

**MAXDEPTH**

Diese Funktion gibt die Höhe eines Knotens anhand des Maximums der rechten und linken Höhe eines Teilbaumes zurück. Potentiell hat ein Knoten eine „rechte Höhe“ als auch eine „linke Höhe“ daher initialisieren wir hierzu zwei variablen „right“ und „left“ jeweils mit dem Wert 1. Sollte es an der rechten Seite des an die Funktion übergeben Knotens einen weiteren Knoten geben so wird auch dieser der Funktion MAXDEPTH übergeben und das Ergebnis dieses Funktionsaufrufes zur variable „right“ hinzuaddiert. Ebenso verhält es sich mit der linken Seite des an die Funktion übergebenen Knotens. Hier wird das Ergebnis natürlich der Variable „left“ hinzuaddiert. Wie oben schon erwähnt wir am Ende der Funktion das maximum von „right“ und „left“ zurückgegeben.