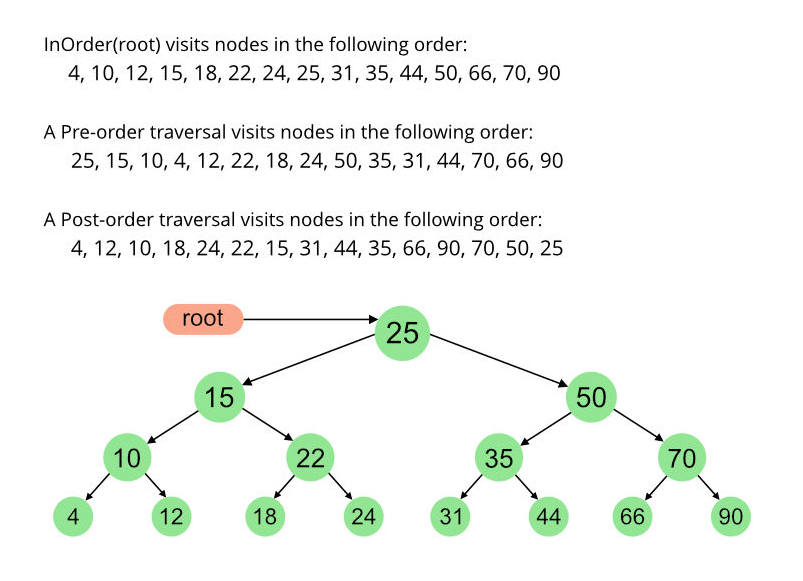
1 Darstellung von Binären Bäumen

**a) Sequentieller Baum in verkettete Darstellung:**  
Hier wird mit einer transform Funktion die sequentielle Darstellung, mithilfe der level by level methode, in eine verkettete Darstellung umgewandelt.

**b) Auf der verketteten Darstellung sind folgende Funktionen zu realisieren:**

**void destroy(Node\_t\*\* ppRoot);**  
Freigabe des gesamten Speicherplatzes des Baums d.h. den Speicherplatz den die Knoten belegen. Hier wird rekursiv jeder Knoten nacheinander gelöscht.  
  
**void printPreOrder(Node\_t\* pRoot)**   
Ausgabe des Baums am Bildschirm in PreOrder

**void printInOrder(Node\_t\* pRoot)**Ausgabe des Baums am Bildschirm in InOrder

**void printPostOrder(Node\_t\* pRoot)**Ausgabe des Baums am Bildschirm in PostOrder

**int countLeaves(Node\_t\* pRoot)**Zählen der Blätter des Baums. Hier gehen wir rekursiv in die Linken und Rechten Nodes bis wir die blätter gefunden haben.

**int height(Node\_t\* pRoot)**Bestimmen der Höhe des Baums. Hier gehen wir rekursiv in die Tiefe des Baumes und bestimmen ganz unten dann einmal die Tiefe des linken Teils und einaml des rechten Teils.

**Testfälle**:  
siehe main.c file.

C-Program code: Siehe beigelegtes binaryTree.c/.h file.

2 Binärer Suchbaum

Hier bauen wir entweder mit der insert funktion oder mit einem Sortiertem Array den binären Suchbaum auf.

**Successor:**  
Hier wird recursiv der Nachfolger für einen bestimmten Wert gesucht. Wenn der rechte Teilbaum nicht null ist dann ist der Nachfolger das am weitesten links stehende Kind des rechten Teilbaums oder das rechte Kind selbst.

**Predecssor:**Hier wird recursiv der Vorgänger für einen bestimmten Wert gesucht. Wenn der linke Teilbaum nicht null ist dann ist der Vorgänger das am weitesten rechts stehende Kind des linken Teilbaums oder das linke Kind selbst.

**Testfälle**:  
siehe main.c file.

C-Program code: Siehe beigelegtes binarySearchTree.c/.h file.