C++ Compact Course

Lab Sheet 3

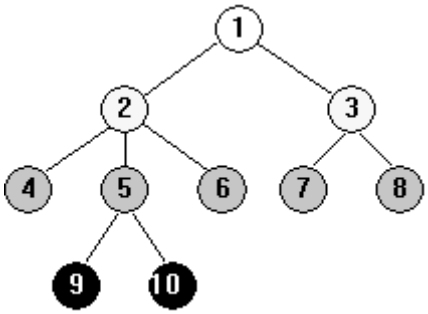
Aufgabe 1: Bäume

Erstellt eine Klasse *Tree*, die einen Baum darstellt. Die Klasse hält immer eine Referenz auf die Wurzel des Baumes bereit. Der Baum selbst besteht aus Elementen der Klasse *TreeNode* (auch die Wurzel).

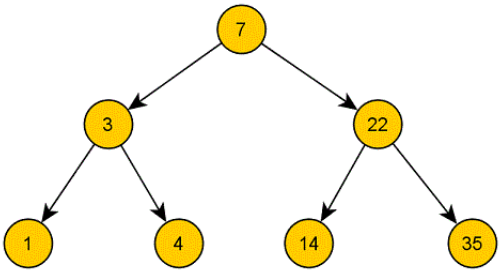
Ein *TreeNode* weiß immer, an welchem anderen *TreeNode* er hängt *(m_parent*), kann weitere *TreeNodes* haben die an ihm hängen (*m_children*) und hat einen Integer Wert (*m_value*).

Verwendet intelligente Zeiger soweit möglich.

1. Erstellt einen Baum, der wie folgt aussieht:



2. Schreibt eine Funktion um euren Baum auf der Console auszugeben.
3. Wandelt euren Baum in einen Binärbaum um. Limitiert dazu die Anzahl der Kinder auf zwei.
4. Schreibt in der Klasse *Tree* eine Funktion, die es erlaubt einen Integer Wert (und somit einen neuen Knoten) in eurem Binärbaum einzusortieren. Dabei soll im Binärbaum immer gelten: Der "linke" Kindknoten eines Knoten ist kleiner als der Knoten und der "rechte" Kindknoten ist größer.



5. Sortiert die Zahlen aus *rand_9.txt* (Aufgabenblatt 01, Aufgabe 1) in euren Binärbaum.

Extra:

1. Modifiziert euren Binärbaum so, dass ihr einen ausgeglichenen Baum (AVL-Baum) erhaltet. In einem AVL-Baum darf sich für jeden Knoten die Höhe der zugehörigen Teilbäume höchstens um eins unterscheiden.

Aufgabe 2: Operator Overloading

1. Überladed `operator>>` für die Ausgabe eurer Bäume.
2. Erstellt zwei Klassen *Matrix* und *Vector* und implementiert `operator+`, `operator-` und `operator*`.

Extra:

1. Schreibt einen Algorithmus, der zu einer gegebenen Matrix A die Inverse A^{-1} berechnet. Testet euer Ergebnis indem ihr $A \cdot A^{-1}$ berechnet.