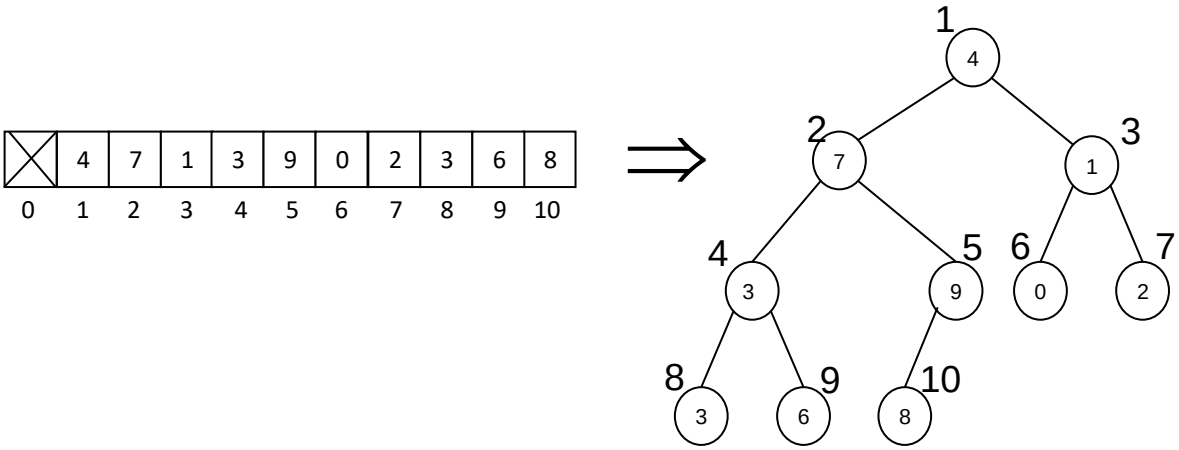
Elementare Algorithmen und Objektorientierte Programmierung

Übung 6

Auszuarbeiten bis 06.05.21

1. Darstellungsformen von Binären Bäumen (5 + 6 Punkte)

a) Gegeben ist ein vollständiger binärer Baum zur Speicherung von Ganzzahlenwerten. Dieser Baum in **sequentieller Darstellung** soll mit Hilfe der Funktion *Transform* in eine **verkettete Darstellung** umgeformt werden.




```
typedef int ItemType;
typedef struct Array {
      int nrOfValues;
      ItemType* pValues;
} Array_t;

typedef struct Node {
      ItemType value;
      struct Node* pLeft;
      struct Node* pRight;
} Node_t;
bool transform (Array_t* pTree, Node_t** ppRoot);
```

Die Daten für die sequentielle Darstellung sollen aus einer Datei eingelesen werden. (Geben Sie auch die Testdateien mit ab!)

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.
- `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
- `int height(Node_t* pRoot)`
  Bestimmen der Höhe des Baums

2. Binärer Suchbaum (9 Punkte)

Entwickeln Sie die folgenden Funktionen für einen *Binären* Suchbaum, in dem Ganzzahlenwerte gespeichert:

- **Einfügen in einem Binären Suchbaum**
  Alle Nachfolger im *linken Unterbaum* haben *kleinere* oder *gleiche* Schlüsselwerte als der Knoten, alle Nachfolger im *rechten Unterbaum* haben *größere* Schlüsselwerte.
- **Bestimmen des Successors (Nachfolger) für einen gesuchten Wert**
  Der Nachfolger eine Knotens *K* ist der Knoten mit dem nächst größeren Integer-Werts
- **Bestimmen des Predecessors (Vorgänger) für einen gesuchten Wert**
  Der Vorgänger eine Knotens *K* ist der Knoten mit dem nächst kleineren Integer-Werts

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

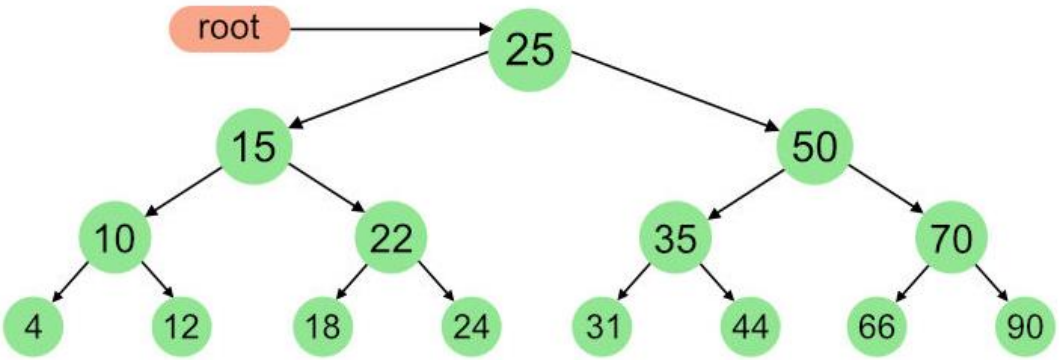
void printInOrder(Node_t* pRoot)
Ausgabe des Baums am Bildschirm in InOrder

void printPostOrder(Node_t* pRoot)
Ausgabe des Baums am Bildschirm in PostOrder

InOrder(root) visits nodes in the following order:
4, 10, 12, 15, 18, 22, 24, 25, 31, 35, 44, 50, 66, 70, 90

A Pre-order traversal visits nodes in the following order:
25, 15, 10, 4, 12, 22, 18, 24, 50, 35, 31, 44, 70, 66, 90

A Post-order traversal visits nodes in the following order:
4, 12, 10, 18, 24, 22, 15, 31, 44, 35, 66, 90, 70, 50, 25



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.