Homework: Advanced Tree Structures - Part I

This document defines the **homework assignments** for the ["Data Structures" course @ Software University](). Please submit a single **zip** / **rar** / **7z** archive holding the solutions (source code) of all below described problems.

Problem 1. AVL Tree

Implement an **AVL tree** by following the guidelines from the [lab document](). The tree should support only **insertion** and **search** operations. Make sure all unit tests pass before you continue.

Use your AVL tree implementation for the next exercises.

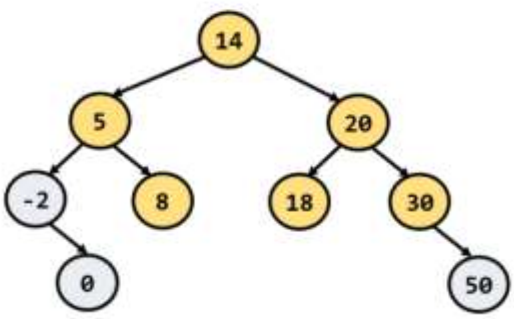
Problem 2. Range in Tree

Implement a **`Range(T from, T to)`** method in your AVL tree for extracting all elements in a given interval (inclusive). The elements should be returned in **ascending order**.

The input will consists of 2 lines:

- The first line holds the **elements** to be inserted (in the order given).
- The second line holds the **interval**.

The elements in range should be printed.

Input	Output	Tree Structure
20 30 5 8 14 18 -2 0 50 50 4 34	5 8 14 18 20 30	
5 40 3 8 2 2 2 1 0 50 80 33 -70 8 40	8 33 40	-
0 0 -10 20 3 4 5 6 7 8 9 10 11 12 13 21 10000	*(empty)*	-

Hints (Click on the arrow to show)

- Use **In-Order DFS** to traverse the tree in ascending order.
- Visit only the nodes which might contain values in the specified range.

Problem 3. * Tree Indexing

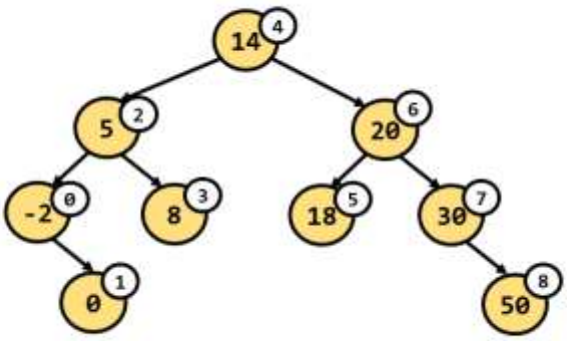
Implement an **indexer** for accessing elements in the tree just like in a list (e.g. **`tree[0]`**, **`tree[5]`**, etc.).

The smallest element has index **`0`**. The largest elements has index **`Count - 1`**. Validate the index for correctness.

The input will consists of several lines:

- The first line holds the **elements** to be inserted (in the order given).
- The next lines will hold the indices.

For each index you must **print its corresponding element** in the tree. If the index is invalid, print "**Invalid index**".

<table>
<tr><th>Input</th><th>Output</th><th>Tree structure</th></tr>
<tr><td>20 30 5 8 14 18 -2 0 50 50
5
2
3
1
-3
9</td><td>18
5
8
0
Invalid index
Invalid index</td><td>
</td></tr>
</table>

Hints (Click on the arrow to show)

- Modify the AVL **Node<T>** class to hold property **Count** (all nodes in its own subtree).
  - Whenever a new node is inserted, its Count is **1**. The retracing should **increase the Count** of all predecessor nodes in the insertion path.
  - When rotations are performed the **Count** should be modified according to the new children using the formula **`node.Count = node.Left.Count + 1 + node.Right.Count`**.
  - You will have to **change the retracing loop** - e.g. we stop modifying balance factors after a rotation, but we must always continue to the root to change the **Count** of all predecessor nodes.
- Indexers in C# are defined like this:

```
public T this[int index]
{
    get
    {
        throw new NotImplementedException();
    }
}
```

- The algorithm for **finding element by index** in a binary search tree is described here: http://stackoverflow.com/a/2329236
- Make sure the new functionality does not break the old one! (Rerun the unit tests from the AVL tree lab)