# Exercise: AVL Trees and AA Trees

This document defines the lab for ["Data Structures – Advanced (C#)" course @ Software University](https://softuni.bg/trainings/4273/data-structures-advanced-with-csharp-november-2023).

## AVL Tree Insertion

You are given a skeleton that supports the following operations:

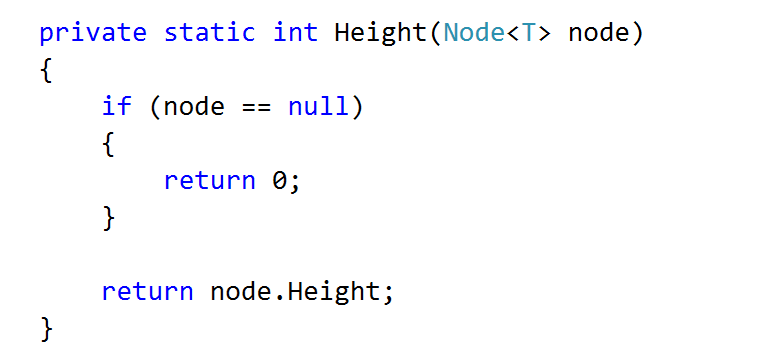
* **Node<T> Root** returns the root of the AVL tree
* **bool Contains(T** **item)** checks if an element exists
* **void EachInOrder(Action<T> action)** performs an action in order on each element
* **void Insert(T item)** inserts an item into the tree

Your task is to balance the tree after each insertion.

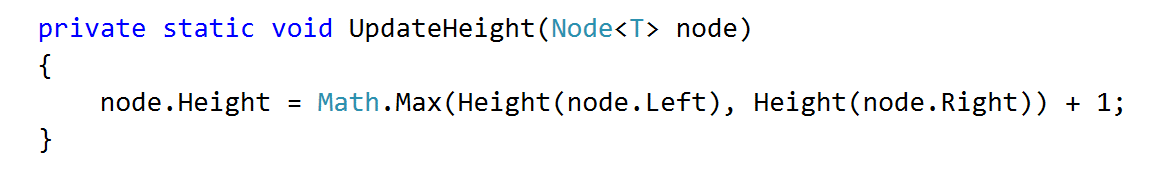
### Height

First, you should update the height of all nodes along an insertion path

You will need a method to find a node's height



And a method to update a node's height



Consider when it is appropriate to update the height of a node

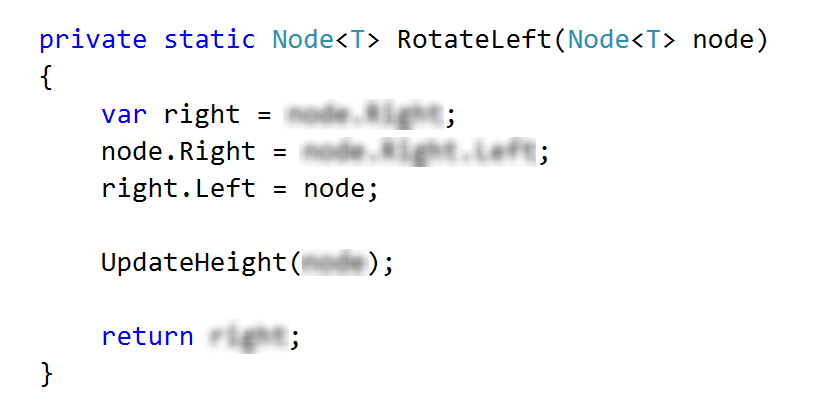


Check if Height tests pass



### Rotations

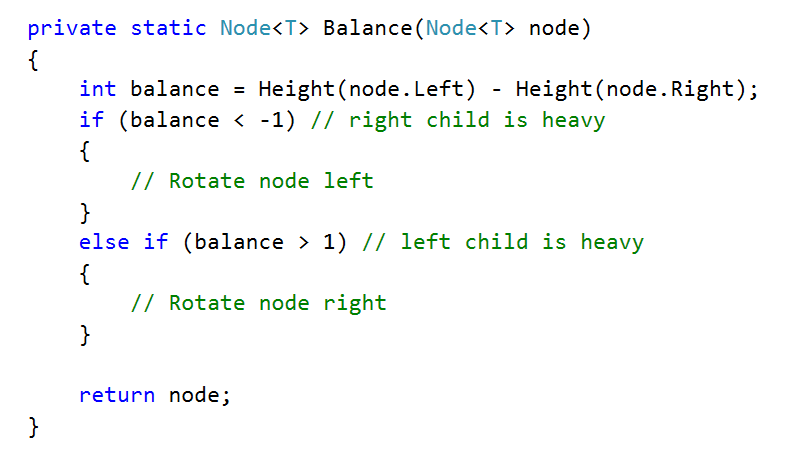
If you find it difficult to imagine the links that need to be updated in a rotation, refer to the presentation



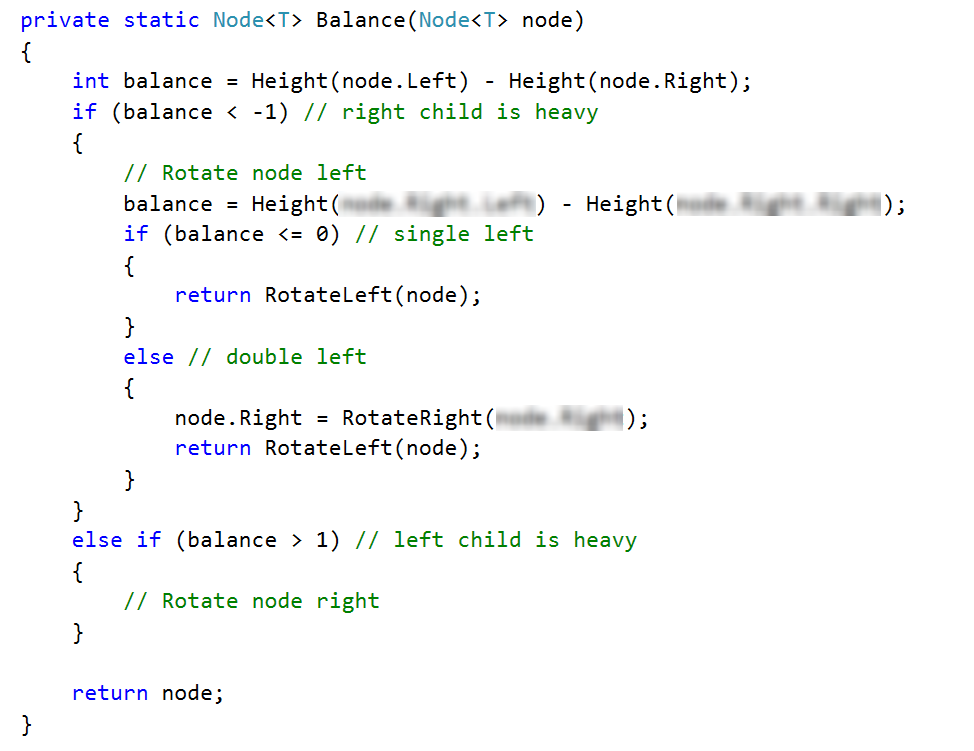
The right rotation is analogous.

### Balancing

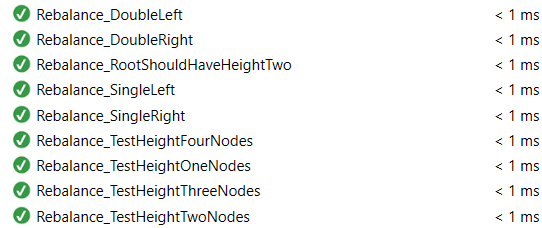
Start by creating the method



The first case (first if statement)



Right rotation is analogous to the left. Make sure that all tests pass:



Congratulations, you have completed the insertion for AVL Trees.

## \* Implement AVL Tree Deletion

Extend your AVL Tree to support:

* void DeleteMin() 🡪 deletes the minimum element (balances the tree if necessary)
* void Delete(T item) 🡪 deletes the given element (balances the tree if necessary)