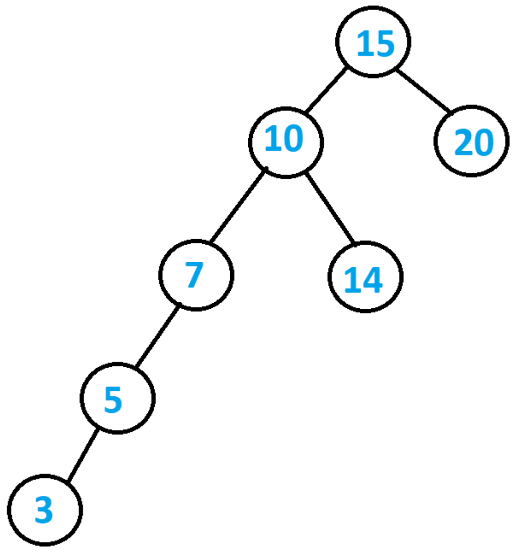
# Pre-Lab Assignment 2

## Instructions

Perform the specified rotations on the given trees below. Some trees require 2 rotations, others may require more. Rotations must be done in the order they are listed.

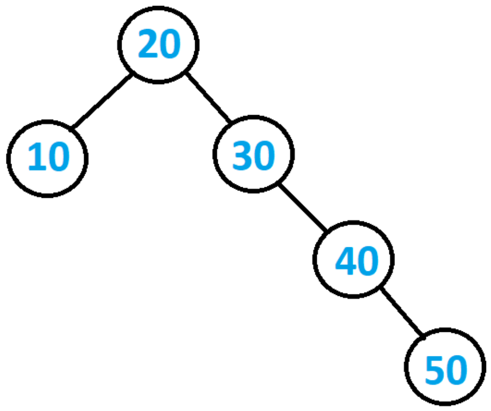
**Tree #1 Rotations:**

- **Right Rotation**(15)
- **Right Rotation**(7)



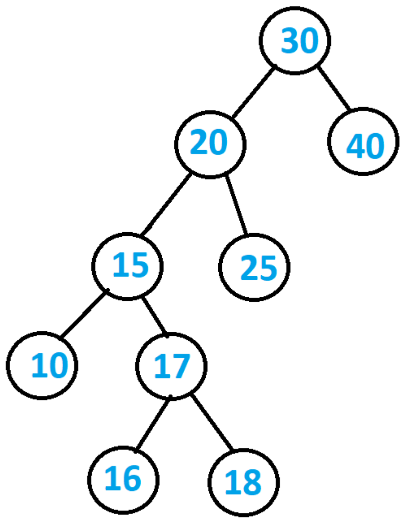
**Tree #2 Rotations:**

- Left Rotation(20)
- Insert(25)
- Insert(22)
- Insert(23)
- Left Rotation(20)
- Right Rotation(30)



**Tree #3 Rotations:**

- **Right Rotation(30)**
- **Left Rotation(15)**



## Submission

Upload to Canvas this file with the completed rotations, trees must be shown after each rotation. Each exercise is worth 5 points, a total of 15/100 for the entire lab.

**Deadline:** on Canvas.

# Lab Assignment 2

CS 302 – Advanced Data Structures and File Processing

## Problem

You are given two non-empty binary search tree $T_1$ and $T_2$. $T_1$ and $T_2$ store the same keys. The structure of both trees, however, is different. Implement an algorithm that uses rotations on $T_1$ to make it equivalent to $T_2$. That is, both trees should have identical structure. Note that you are only allowed to use rotations and only on $T_1$; you are not allowed to modify the trees in any other way. **The implementation must be iterative.**

## Implementation

You are given two files (which you can download from canvas): *Lab2.java* and *BST.java*. The file *Lab2.java* generates test cases, performs the tests, and outputs the results. The file *BST.java* partially implements a binary search tree and contains the function `problem`; implement your solution in that function. **Do not make any changes outside of that function; such changes will be undone.** Do not output anything to the terminal. The class `BST` also contains the functions `rotateL`, `rotateR`, `find`, as well as functions for in- and pre-order. Feel free to use these functions in your implementation.

The program already implemented in the file `Lab2java` randomly generates test cases. The seed of the random number generator is set to ensure the same test cases whenever to program is executed. Note that the purpose of the tests is for you to avoid major mistakes. **Passing all given tests *does not* imply that your algorithm is correct, especially that is has the expected runtime.**

## Submission

For your submission, upload the file *BST.java* with your implementation to canvas. This part is worth 85/100 for the overall lab.

This is an individual assignment. Therefore, a submission is required from each student.

**Deadline:** on Canvas