Data Structures
Term Project 3
Due: Dec. 12th by 23:59 in LMS

Implement your solution for the following problem using Java. Use only the constructs you learned in class.

1. Implement Binary Search Tree (BST) class. (50 points)
2. In you main function, have an array that contains 10 numbers, randomly generated between 1 and 100. Insert all the numbers in the order they appear in the array into your BST. Draw your BST on a piece of paper and check how balanced it is. (5 points)
3. Do an **in-order traversal** to extract the sorted sequence of number into another array of the same size. (5 points)
4. Rebuild a **perfectly balanced BST** from that sorted array. Draw your BST on a piece of paper and check how balanced it is. (5 points)
5. Assume the definition that your tree is balanced if the height of the left subtree and right substree difference is 1, that is, $|\text{height(leftSubtree)} - \text{height(rightSubtree)}| \leq 1$, and the subtrees are also balanced (recursive definition?). Implement this definition and check if your BST's in (2) and (4) are balanced. (15 points)
6. Implement the following metrics which measures how badly a BST could be: (20 points)
    1. Compare the tree height to the *ideal balanced height*:

$$h_{\text{ideal}} = \lfloor \log_2(n) \rfloor$$

$$imbalance = \frac{h_{\text{actual}}}{h_{\text{ideal}}}$$

Examples:

- Ratio = **1.0** → perfectly balanced
- Ratio = **2.5** → somewhat skewed
- Ratio = **5–10** → very badly skewed (almost a linked list)

    2. Sum up the imbalance at *each* node:

$$imbalance\_score = \sum_{v \in nodes} |\, height(left(v)) - height(right(v)) \,|$$

- Score = 0 → perfectly balanced
- Higher score → many badly unbalanced spots

This gives a "total unbalance energy" of the tree.

**Deliverables:**

Turn in your .java code with your output and comments at the end of your program to LMS folder for Project 3 submission.