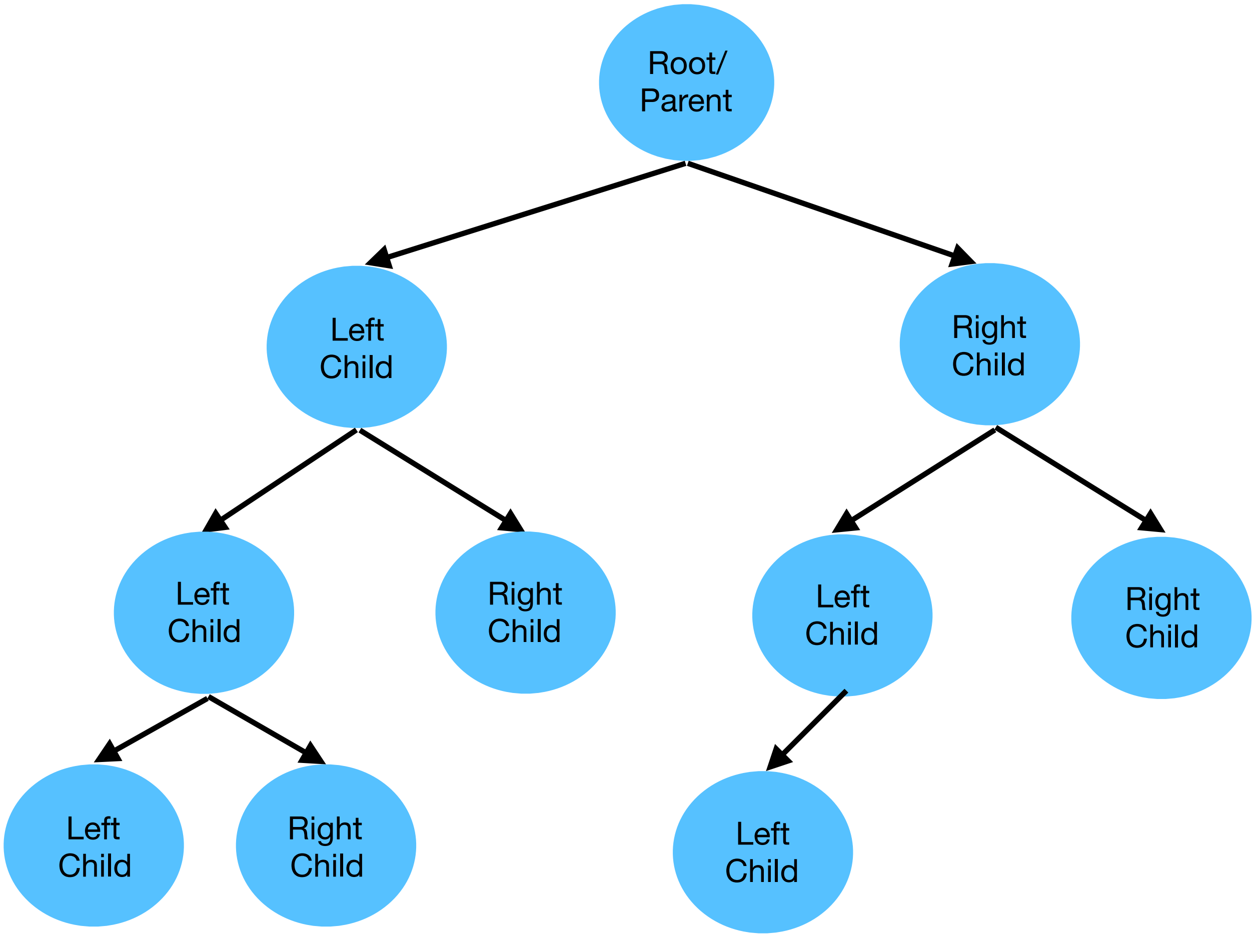
Project 3: Binary Search Tree

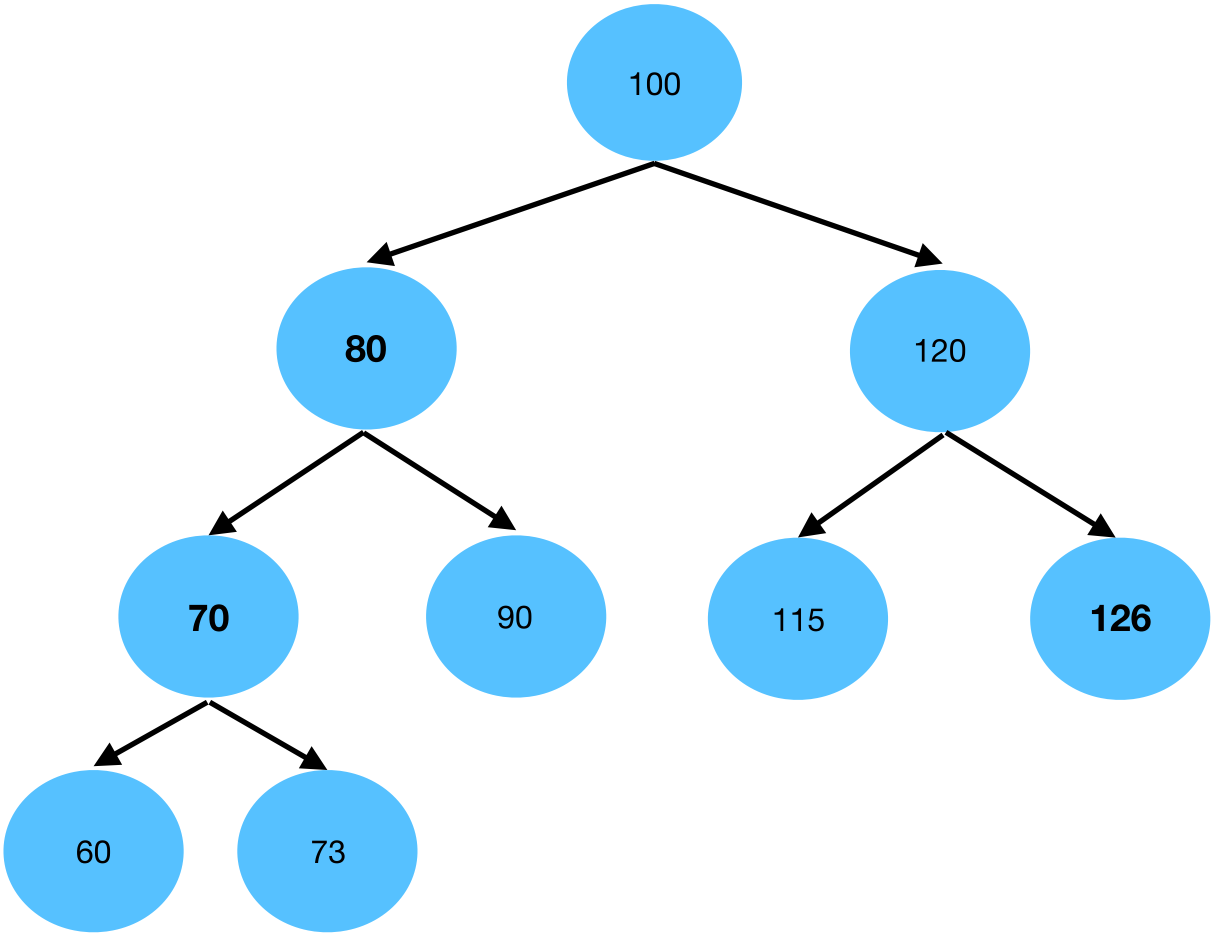
Juanjuan Xia

Part 1: Practice Question

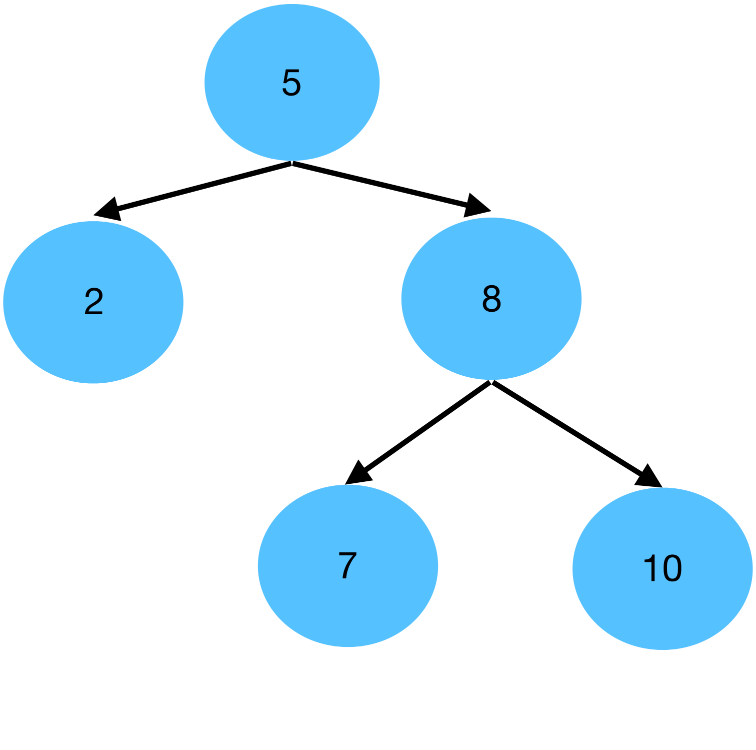
1. Draw a binary tree with 10 node. Only consider the structure of the tree, using the diagram above as a reference. Disregard the values stored in the tree for this question.



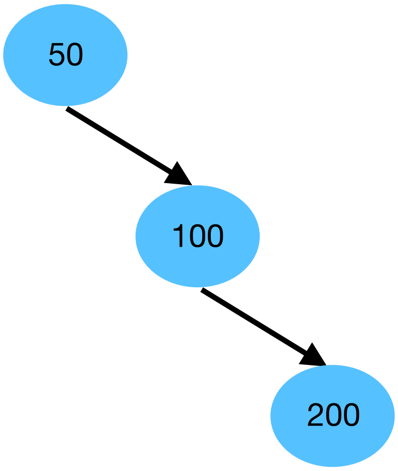
2. Complete the following Binary Search Tree, adding the values 126, 80, and 70. Use the rules described in the section above to determine where each value should be placed in the hierarchy.



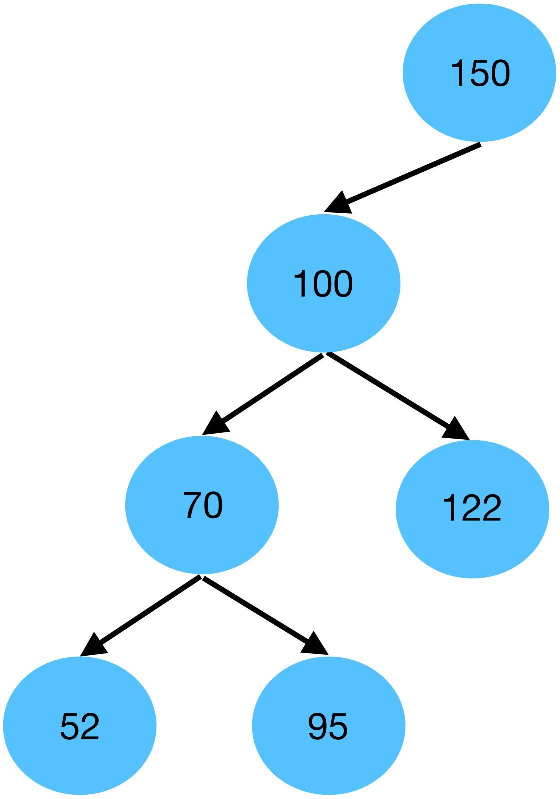
3. Draw a Binary Search Tree which stores the values 5, 8, 2, 10, and 7. Insert the values into the tree in that order. Use the rules described in the section above to determine where each value should be placed in the hierarchy.



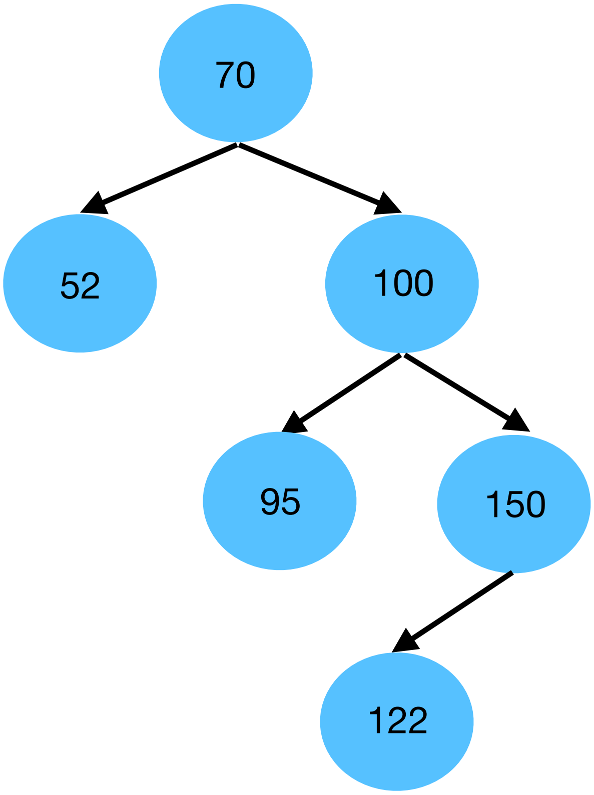
4. Show the result of a Right Rotation performed on the following tree. In other words, rotate the following tree to the right. Remember that we spent approximately two hours doing examples of these rotations in class, and an example implementation of this operation is given in the tree demo posted on iLearn.



5. a) Show the result of a Left Rotation performed on the following tree.

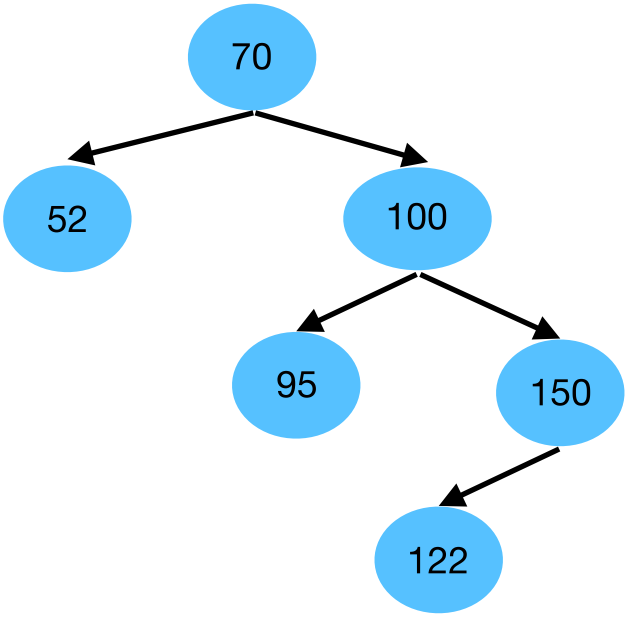


5. b) Show the result of a Right Rotation performed on the tree shown in the above question(5.a).

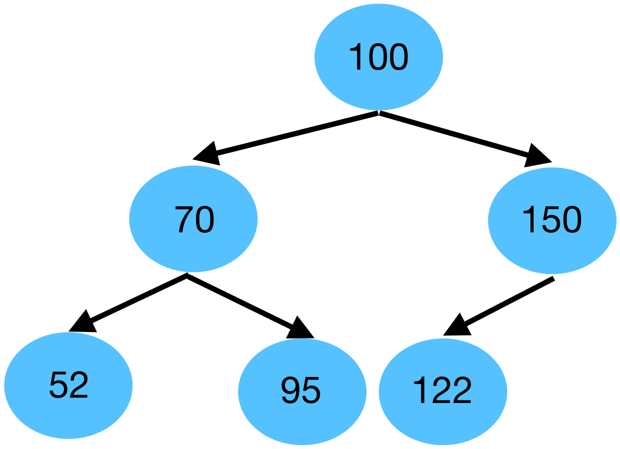


6. a) Assume the tree shown in question 5 is rotated Left, then rotated Right. How will the resulting tree differ from the original?

1) **After left rotation:** the diagram showed in 5.b.



2) **then after right rotation:** the resulting tree is the same as the original.



6. b) Will the order and position of the nodes change after performing a Left rotation followed by a Right rotation?

1) performing a Left rotation: the order and position of the nodes change.
2) followed by a Right rotation: the order and position of the nodes change again.

Finally, the order and position of the resulting nodes are the same as the original.

7. Write the value of each node visited when performing a pre-order traversal of the following tree:

1) **100** 2) **70** 3) **50** 4) **90** 5) **120** 6) **115** 7) **150**

8. Write the value of each node visited when performing an in-order traversal of the tree above.

1) **50** 2) **70** 3) 90 4) 100 5) **115** 6) **120** 7) **150**

Part 2: Implementation Question

1. Using the Node class defined above, write an algorithm to insert a new Node into a Binary Search Tree. Assume the root of the tree is stored in a variable called “root”. Use the following function signature when writing your code: …

```
void insert (Node newNode) {
        if (root == null) {
                root = new Node(newNode.data);
        } else if (newNode.data < root.data) {
                if (root.left == null) {
                        root.left = new Node(newNode.data);
                } else {
                        root.left.insert(newNode);
                }
        } else if (newNode.data >= root.data) {
                if (root.right == null) {
                        root.right = new Node(newNode.data);
                } else {
                        root.right.insert(newNode);
                }
        }
}
```

2. Write a function which takes the root node of a Binary Search Tree as a parameter, and prints the values stored in the tree in order from smallest to largest.

```
public void inorder (Node<Integer> root) {
        if (root != null) {
                inorder(root.left);
                System.out.print(root.data + “, ”);
                inorder(root.right);
        }
}
```

3. Write a function which takes the Root node of a Binary Search Tree and an integer value called "target" as parameters. The function should search the tree for the node storing the "target" value and return the Node object. If the given value is not found in the tree, the function should return null.

```
public Node search (Node root, int target) {
        if (root == null || root.data == target) {
                return root;
        }

        if (root.data > target) {
                return search(root.left, target);
        }

        return search(root.right, target);
}
```

4. Write pseudocode to perform a left rotation and right rotation on a Binary Search Tree. Assume the root node of the tree is stored in a variable named "root". Remember that "pseudocode" is simply an less formal description of each operation needed to perform an algorithm. Each step should be clearly defined. An example of pseudocode for finding the average value in an array is given below: …

1) Left Rotation

```
public void rotateLeft(Node n)
        if n.right == null  then return

        Node oldRight = new Node(n.right.data)
        oldRight.left = n.right.left
        oldRight.right = n.right.right

        n.right = n.right.left

        if n.root == null then root = oldRight
        else if n.root.left == n then n.root.left = oldRight
```

```
    else n.root.right = oldRight

    oldRight.left = n
```

2) Right Rotation

```
protected void rotatedRight(Node n)
    if n.left == null  then return

    Node oldLeft = new Node(n.left.data)
    oldLeft.left = n.left.left
    oldLeft.right = n.left.right

    n.left = oldLeft.right

    if n.root == null then root = oldLeft
    else if n.root.left == n then n.root.left = oldLeft
    else n.root.right = oldLeft

    oldLeft.right = n
```