Week 11 Exercises Submit

Sandra Batista

1.1–1.2

Exercise 1: Creating a Balanced BST

1. Give an insertion order of the numbers 1, 2, 3, 4, 5, 6, 7 Such that the resulting BST is balanced and draw the BST.

- Recall that a Balanced D-ary Tree: Tree where subtrees of any node differ in height by at most 1
- |Height(right subtree of root) – Height(left subtree of root)| <= 1

Exercise 2: BST Find

Starter code:
https://github.com/sandraleeusc/csci104_fall2020_lecture/blob/master/bst_exercises.cpp

1. Write a recursive function to **find** a value in a BST.
2. Trace the function on the BST in main for a value not in the tree such as 5 and a value in the tree such as 3

Exercise 2.5: Extra Recursion

Trace the following code on binary tree of your choice. Does the binary tree matter? Does it matter if it is a BST? (Node struct is same as in other binary tree examples…)

```
int mystery(shared_ptr<Node> x) {
        if (x == nullptr) return 0;
        else return max(mystery(x->left), mystery(x->right));
}
```

(Source: https://www.cs.princeton.edu/courses/archive/fall02/cos126/exercises/trees.html from a Week 10 Question Bank Submission)

Exercise 3: BST FindAllInRange

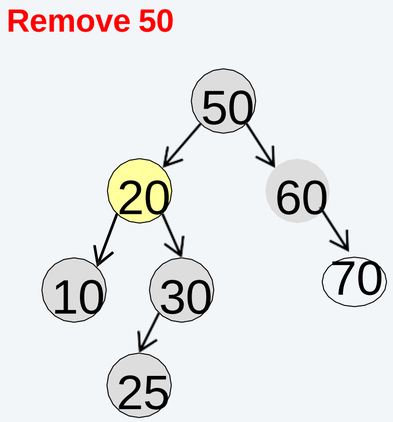
Starter code:

https://github.com/sandraleeusc/csci104_fall2020_lecture/blob/master/bst_exercises.cpp

1. Write a function, **findAllInRange**, that given a root to a balanced binary search tree and two values, k1 and k2, prints out all the keys in the tree that are greater than or equal to k1 and less than or equal to k2 from smallest to largest. Your implementation must be recursive and run in O(log(n) + v) where v is the number of key values in the range and n is the total number of keys in the tree. You may use helper functions. No loops allowed.
2. Modify the code in main to test your function. You can also trace how your code should behave on your tests.

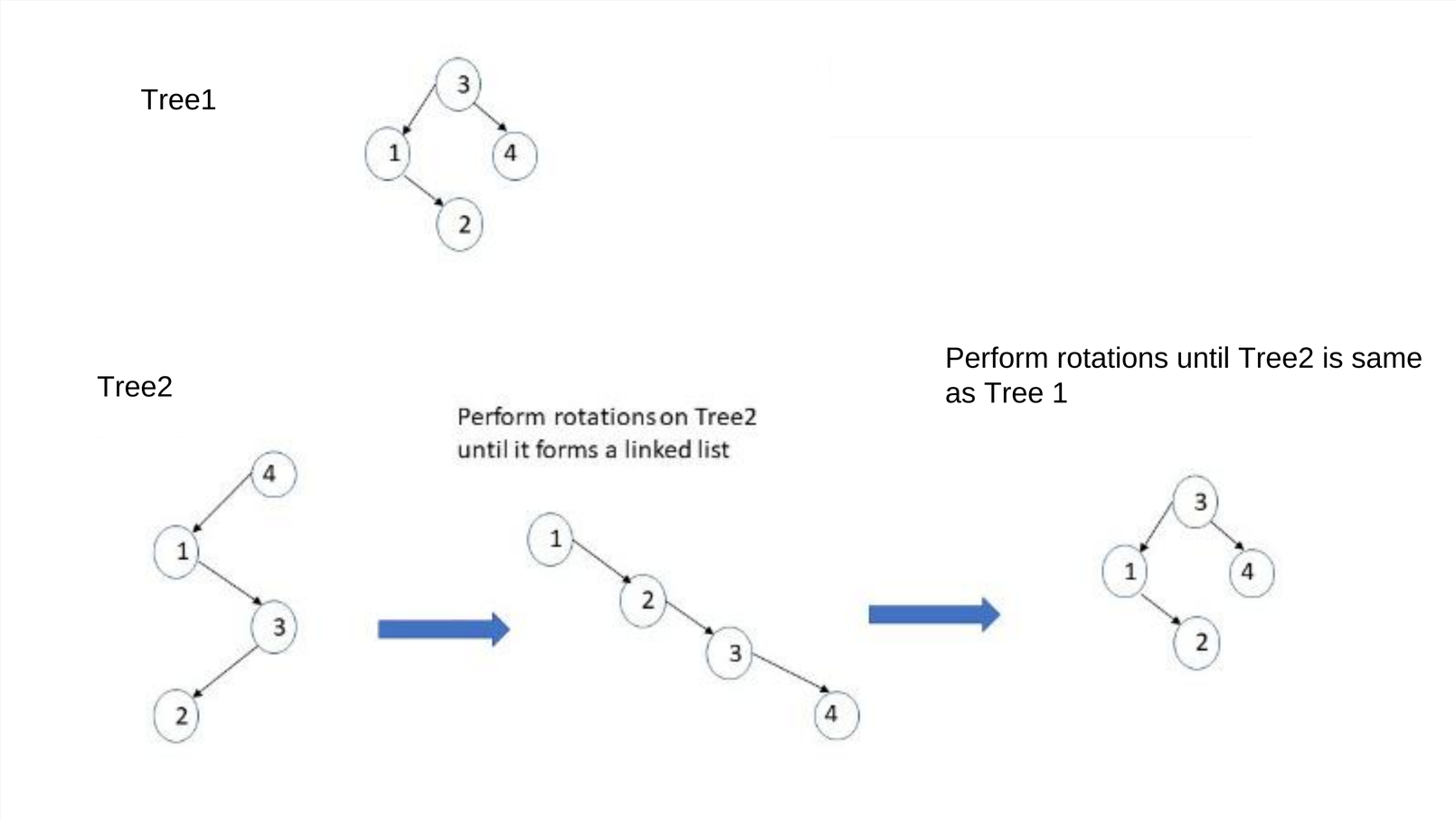
4. Tracing BST Remove

1. In the following tree trace removing 50 by i) swapping with the predecessor and ii) swapping with the successor.
2. What must be considered if neither the successor nor the predecessor are leaf nodes? Can the successor or predecessor have two children? Why or Why not? If so, how does this affect removal?



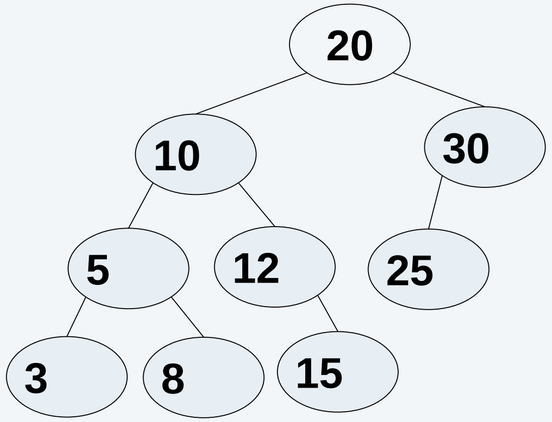
5. Fundamental Theorem of Rotations

What rotations must be performed to transform Tree2 into Tree1? Use rotateRight(z,y) and rotateLeft(x,y) in your answer.



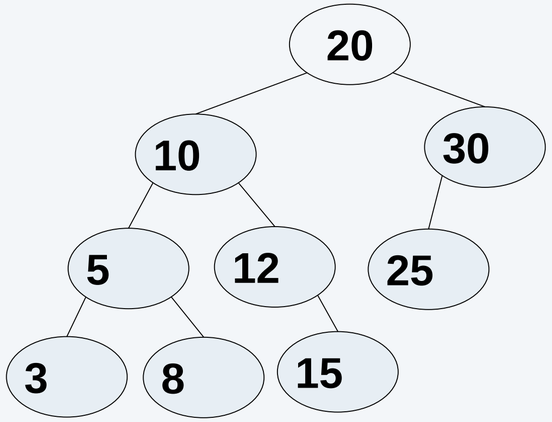
6. AVL Insertion Practice

a) Insert key=28



6. AVL Insertion Practice

b) Insert key=17



6. AVL Insertion Practice

c) Insert key=2