CSC 212 Tutorial #8 Solution
Binary Trees

Problem 1

```
//Implementer
public int countLeafNodes() {
    return countLeafNodesRec(root);
}

private int countLeafNodesRec(BTNode<T> n) {
    if (n == null)
        return 0;
    if (n.left == null && n.right == null)
        return 1;
    return countLeafNodesRec(n.left) + countLeafNodesRec(n.right);
}

//User
public static <T> int countLeafs(BT<T> bt, T e) {
    if (bt.empty()) {
        return 0;
    }
    bt.find(Relative.Root);
    return recCountLeafs(bt);
}

private static <T> int recCountLeafs(BT<T> bt) {
    if (bt.isLeaf())
            return 1;
    else {
        int nb = 0;
        if (bt.find(Relative.LeftChild)) {
            nb += recCountLeafs(bt);
            bt.find(Relative.Parent);
        }
        if (bt.find(Relative.RightChild)) {
            nb += recCountLeafs(bt);
            bt.find(Relative.Parent);
        }
        return nb;
    }
}
```

Problem 2

```
public int countNodesIn(int k) {
    BSTNode<T> n = root;
    while (n != null) {
        if (n.key == k)
            break;
        else if (k < n.key)
            n = n.left;
        else
            n = n.right;
    }
    return countNodesInRec(n);
}

private int countNodesInRec(BSTNode<T> n) {
    if (n == null)
        return 0;
    return 1 + countNodesInRec(n.left) + countNodesInRec(n.right);
}
```

Problem 3

1. Inserting

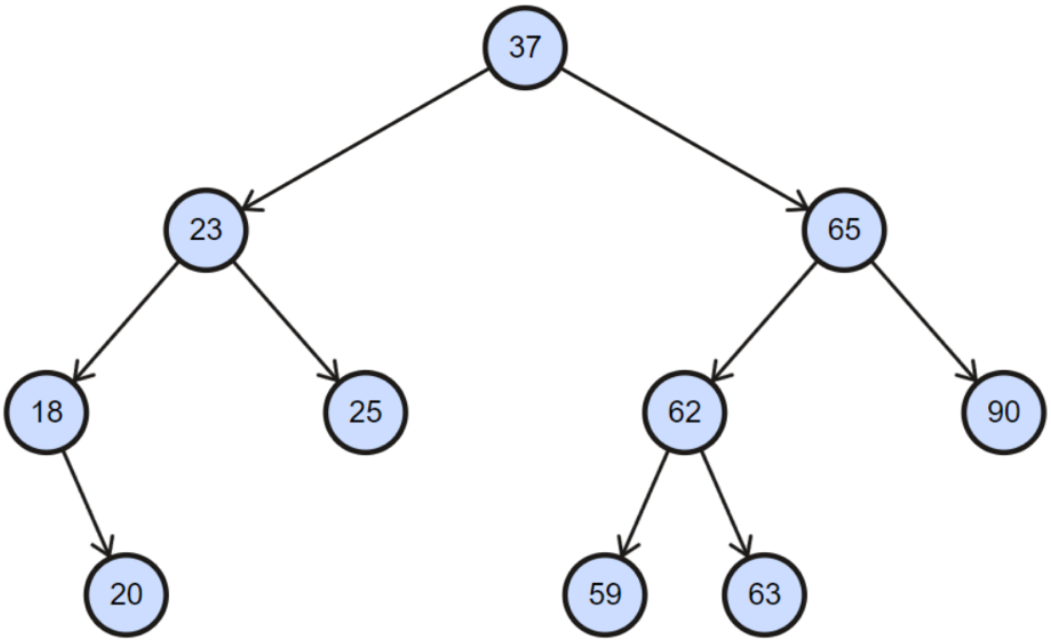

Figure 1: After inserting all keys

2. Removing

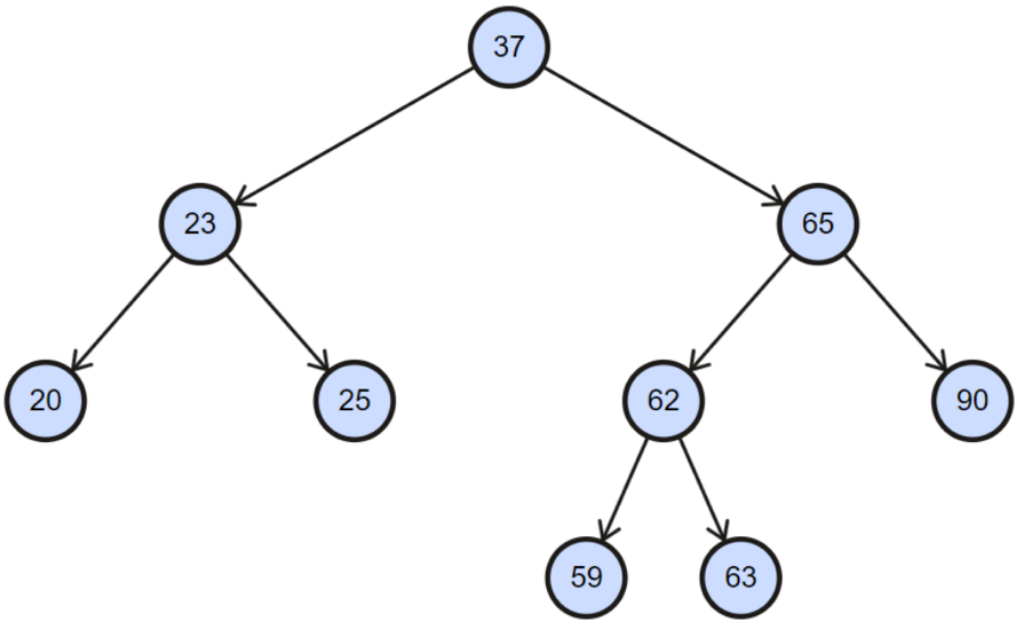

Figure 2: Removing 18 (case 2: having one child)

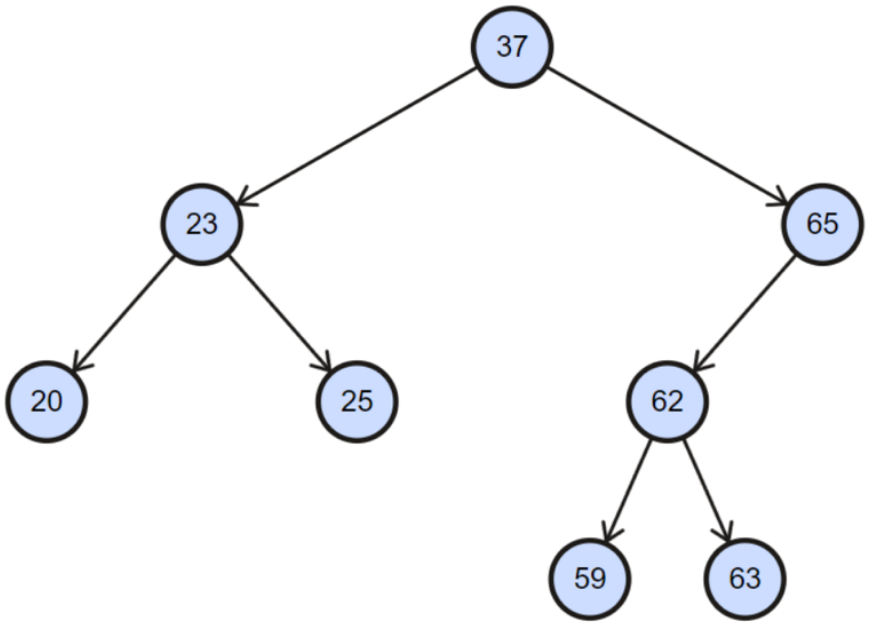

Figure 3: Removing 90 (case 1: having no children)

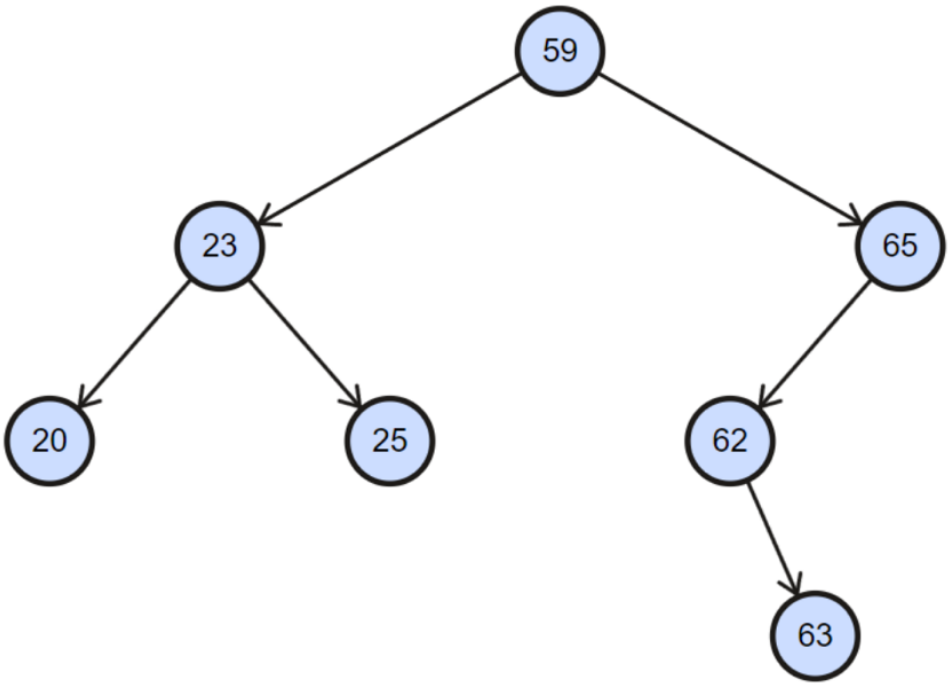

Figure 4: Removing 37 (case 3: having two children; left-most node in the right sub-tree)

3. In-order (left-root-right)