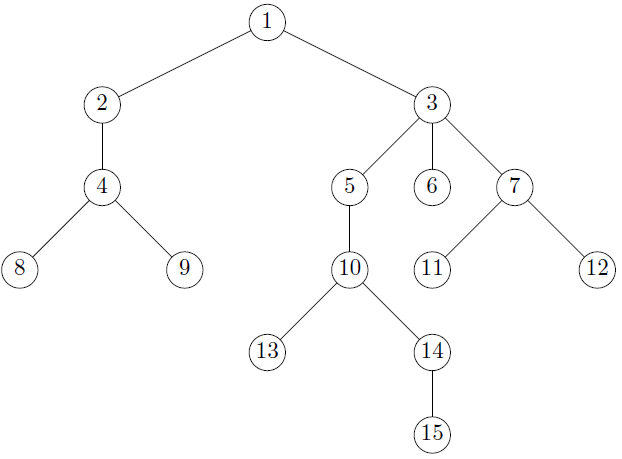
Lab 7

Trees

1. Questions. (10 points, about 0.5 points per question)   
Thanks to Allo & Ullman, Foundations of Computer Sceience, Exercise 5.2.1.



For this tree, what is described by each of the following phrases?

a) The root of the tree: **Node 1**

b) The leaves of the tree: **Nodes 8, 9, 13, 15, 11, 6, and 12.**

c) The interior nodes of the tree: **Nodes 1, 2, 3, 4, 5, 7, 10, and 14**.

d) The siblings of node 6: **5, and 7**

e) The subtree with root 5: **Nodes 5, 10, 13, 14, and 15.**

f) The ancestors of node 10: **Nodes 5, 3, and 1.**

g) The descendants of node 10: **Nodes 13, 14, and 15.**

h) Siblings to the left of node 10: **Nodes 8 and 9.**

i) Siblings to the right of node 10: **Nodes 11 and 12.**

j) The longest path in the tree: **Nodes 1, 3, 5 ,10, 14, and 15.**

k) The depth of node 3: **2**

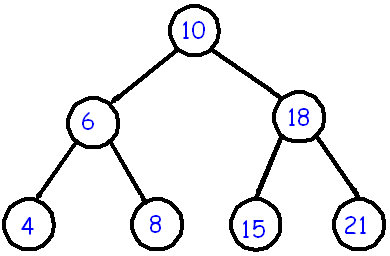
l) The depth of node 13: **5**

m) The height of the tree: **6**

2. Trees in Java. (10 points)

A sample file of work with a TreeNode class is provided with the lab. Code to output a three-level tree to the console is also included.

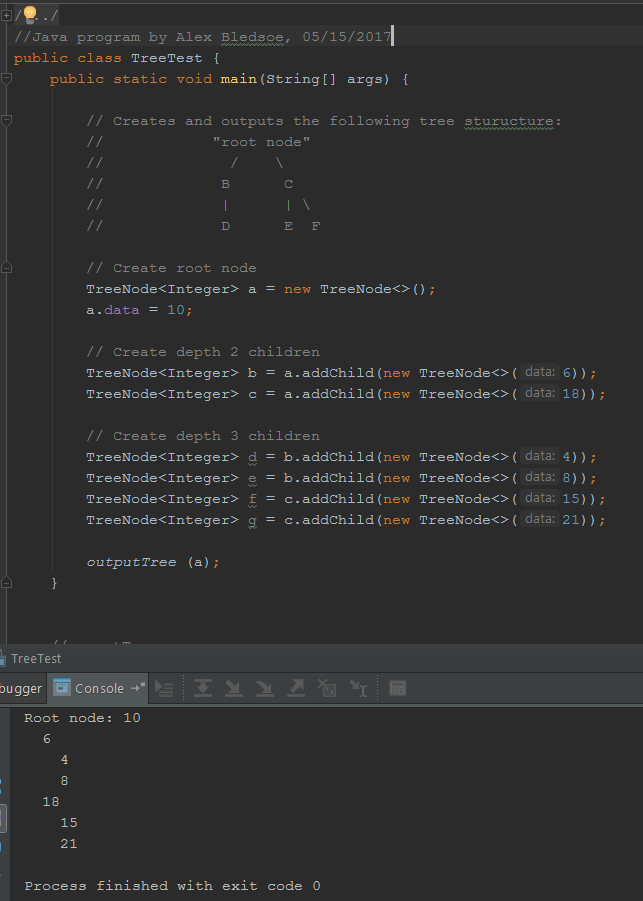
Adapt the provided program to recreate the following tree:



Thanks to: <https://www.cs.cmu.edu/~adamchik/15-121/lectures/Trees/trees.html>

Rubric:  
Student name and today’s date is a comment in the first line of the programs: -5 points if fails  
Screenshot and program code: -5 points if fails   
Tree creation: 3 points per depth level  
Tree output using supplied routine: 1 point

Please paste a screenshot of a successful program run, and copy-and-paste the source code from your main program's .java file, here. You do not need to include the TreeNode.java file.



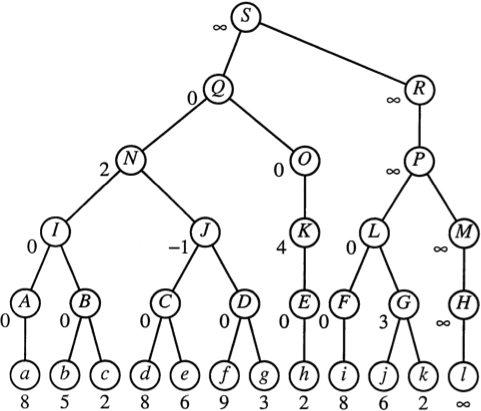
**TreeTest.java:**

//Java program by Alex Bledsoe, 05/15/2017  
// TreeNode class thanks to http://stackoverflow.com/questions/19330731/tree-implementation-in-java-root-parents-and-children  
  
public class TreeTest {  
 public static void main(String[] args) {  
   
 // Creates and outputs the following tree sturucture:  
 // "root node"  
 // / \  
 // B C  
 // | | \   
 // D E F  
   
 // Create root node  
 TreeNode<Integer> a = new TreeNode<>();  
 a.data = 10;  
   
 // Create depth 2 children  
 TreeNode<Integer> b = a.addChild(new TreeNode<>(6));  
 TreeNode<Integer> c = a.addChild(new TreeNode<>(18));  
   
 // Create depth 3 children  
 TreeNode<Integer> d = b.addChild(new TreeNode<>(4));  
 TreeNode<Integer> e = b.addChild(new TreeNode<>(8));  
 TreeNode<Integer> f = c.addChild(new TreeNode<>(15));  
 TreeNode<Integer> g = c.addChild(new TreeNode<>(21));  
   
 *outputTree* (a);   
 }

**TreeNode.java:**

//Java program by Alex Bledsoe, 05/15/2017  
  
import java.util.ArrayList;  
import java.util.List;  
  
// Thanks to http://stackoverflow.com/questions/19330731/tree-implementation-in-java-root-parents-and-children  
// Edited by MCH, 20170503  
  
public class TreeNode<T>{  
 public T data = null;  
 private List<TreeNode> children = new ArrayList<>();  
 private TreeNode parent = null;  
  
 public TreeNode() {  
 }  
   
 public TreeNode(T data) {  
 this.data = data;  
 }  
  
 public TreeNode<T> addChild(TreeNode child) {  
 child.setParent(this);  
 this.children.add(child);  
 return child;  
 }  
  
 public TreeNode<T> addChild(T data) {  
 TreeNode<T> newChild = new TreeNode<>(data);  
 newChild.setParent(this);  
 children.add(newChild);  
 return newChild;  
 }  
  
 public void addChildren(List<TreeNode> children) {  
 for(TreeNode t : children) {  
 t.setParent(this);  
 }  
 this.children.addAll(children);  
 }  
  
 public List<TreeNode> getChildren() {  
 return children;  
 }  
  
 private void setParent(TreeNode parent) {  
 this.parent = parent;  
 }  
  
 public TreeNode getParent() {  
 return parent;  
 }  
}

3. Binary trees (5 points)



Thanks to: <http://jeffe.cs.illinois.edu/teaching/datastructures/2011/Fig/topology-tree.png>

3.1. Why is this a valid binary tree?

**- Every node in the tree has no more than 2 children.**

3.2. Why is this not a "proper" binary tree?

**- Some of the nodes have only one child.**

3.3. What is the left child of node S?

**- Q**

3.4. What is the right child of node S?

**- R**

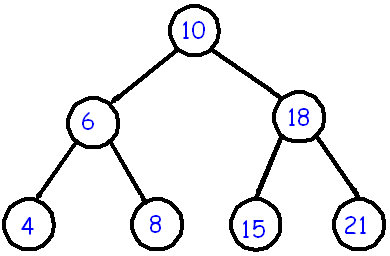
3.5. Is the subtree with root node J a proper binary tree? Why or why not?

**- Yes, because J and all its children in the subtree have 2 children.**

4. In-order and post-order traversal (20 points)

Implement the textbook's method for in-order and post-order traversal. To "process" or "visit" a node means to print the data at that node.

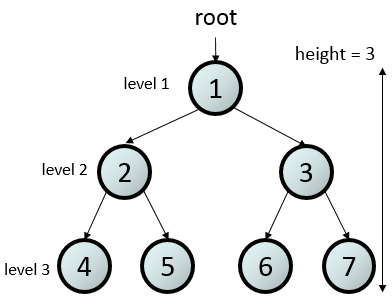
Feel free to reuse the tree and code from problem 2 as a test case.

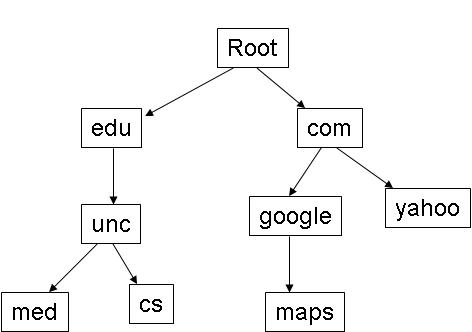
The tree from problem 2: 

Pre-order traversal output: 10 6 4 8 18 15 21

Post-order traversal output: 4 8 6 15 21 18 10

Additionally, create and traverse the following trees:

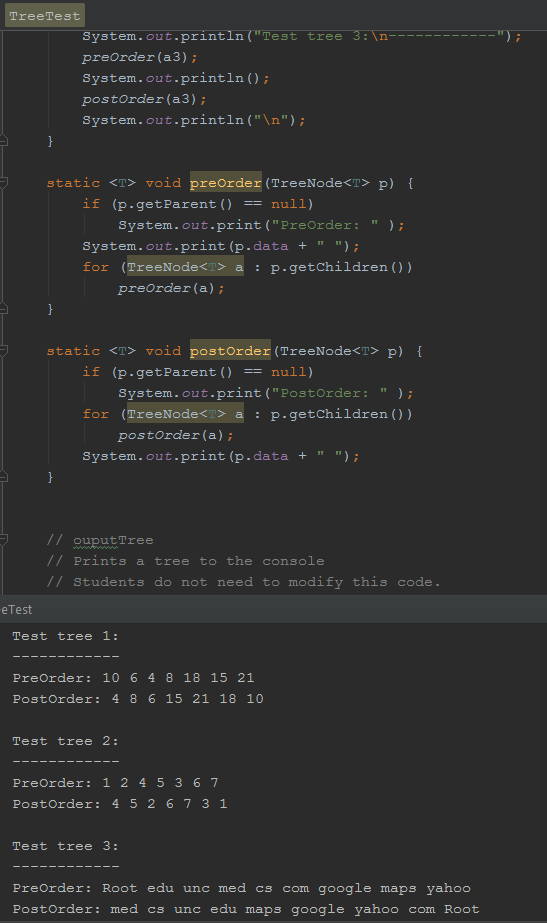
  
Thanks to <https://stanford.edu/class/archive/cs/cs106b/cs106b.1158/images/tree-height-figure.png> .



Thanks to <http://www.cs.unc.edu/~dewan/comp114/f05/recitation/> .

Rubric:  
Student name and today’s date is a comment in the first line of the programs: -10 points if fails  
Screenshot and program code: -15 points if fails   
All lines of program output shown: -10 if fails  
Tree creation for two additional trees: 6 points  
Pre-order traversal and output for all three trees: 7 points  
Post-order traversal and output for all three trees: 7 points

Please paste a screenshot of a successful program run **with all lines of output**, and copy-and-paste the source code from your main program's .java file, here. You do not need to include the TreeNode.java file.



**TreeTest.java:**

// Java program by Alex Bledsoe, 05/23/2017  
  
// TreeNode class thanks to http://stackoverflow.com/questions/19330731/tree-implementation-in-java-root-parents-and-children  
  
public class TreeTest {  
 public static void main(String[] args) {  
  
 //TEST TREE 1  
   
 // Creates and traverses the following tree structure:  
 // 10  
 // / \  
 // 6 18  
 // / \ / \  
 // 4 8 15 21  
   
 // Create root node  
 TreeNode<Integer> a1 = new TreeNode<>();  
 a1.data = 10;  
   
 // Create depth 2 children  
 TreeNode<Integer> b1 = a1.addChild(new TreeNode<>(6));  
 TreeNode<Integer> c1 = a1.addChild(new TreeNode<>(18));  
   
 // Create depth 3 children  
 TreeNode<Integer> d1 = b1.addChild(new TreeNode<>(4));  
 TreeNode<Integer> e1 = b1.addChild(new TreeNode<>(8));  
 TreeNode<Integer> f1 = c1.addChild(new TreeNode<>(15));  
 TreeNode<Integer> g1 = c1.addChild(new TreeNode<>(21));  
  
 System.*out*.println("Test tree 1:\n------------");  
 *preOrder*(a1);  
 System.*out*.println();  
 *postOrder*(a1);  
 System.*out*.println("\n");  
  
 */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* //TEST TREE 2  
  
 // Creates and traverses the following tree structure:  
 // 1  
 // / \  
 // 2 3  
 // / \ / \  
 // 4 5 6 7  
  
 // Create root node  
 TreeNode<Integer> a2 = new TreeNode<>();  
 a2.data = 1;  
  
 // Create depth 2 children  
 TreeNode<Integer> b2 = a2.addChild(new TreeNode<>(2));  
 TreeNode<Integer> c2 = a2.addChild(new TreeNode<>(3));  
  
 // Create depth 3 children  
 TreeNode<Integer> d2 = b2.addChild(new TreeNode<>(4));  
 TreeNode<Integer> e2 = b2.addChild(new TreeNode<>(5));  
 TreeNode<Integer> f2 = c2.addChild(new TreeNode<>(6));  
 TreeNode<Integer> g2 = c2.addChild(new TreeNode<>(7));  
  
 System.*out*.println("Test tree 2:\n------------");  
 *preOrder*(a2);  
 System.*out*.println();  
 *postOrder*(a2);  
 System.*out*.println("\n");  
  
 */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* //TEST TREE 3  
  
 // Creates and traverses the following tree structure:  
 // \_ Root \_  
 // / \  
 // edu com  
 // | / \  
 // unc google yahoo  
 // / \ |  
 // med cs maps  
  
 // Create root node  
 TreeNode<String> a3 = new TreeNode<>();  
 a3.data = "Root";  
  
 // Create depth 2 children  
 TreeNode<String> b3 = a3.addChild(new TreeNode<>("edu"));  
 TreeNode<String> c3 = a3.addChild(new TreeNode<>("com"));  
  
 // Create depth 3 children  
 TreeNode<String> d3 = b3.addChild(new TreeNode<>("unc"));  
 TreeNode<String> e3 = c3.addChild(new TreeNode<>("google"));  
 TreeNode<String> f3 = c3.addChild(new TreeNode<>("yahoo"));  
  
 // Create depth 4 children  
 TreeNode<String> g3 = d3.addChild(new TreeNode<>("med"));  
 TreeNode<String> h3 = d3.addChild(new TreeNode<>("cs"));  
 TreeNode<String> i3 = e3.addChild(new TreeNode<>("maps"));  
  
  
 System.*out*.println("Test tree 3:\n------------");  
 *preOrder*(a3);  
 System.*out*.println();  
 *postOrder*(a3);  
 System.*out*.println("\n");  
 }  
  
 static <T> void preOrder(TreeNode<T> p) {  
 if (p.getParent() == null)  
 System.*out*.print("PreOrder: " );  
 System.*out*.print(p.data + " ");  
 for (TreeNode<T> a : p.getChildren())  
 *preOrder*(a);  
 }  
  
 static <T> void postOrder(TreeNode<T> p) {  
 if (p.getParent() == null)  
 System.*out*.print("PostOrder: " );  
 for (TreeNode<T> a : p.getChildren())  
 *postOrder*(a);  
 System.*out*.print(p.data + " ");  
 }  
}