Problem Set 5

Assigned: Oct. 23
Due: Oct. 30

For problem 3 below, I prefer that you use line drawing software, but, it is OK to draw the answers to problem 3 by hand, take a photo, and upload that as a separate file. Problems 1, 2, and 4 on the assignment should be typeset, as usual.

Problem 1

Show the pre-order, post-order, in-order, and breadth-first search order of the tree below.

![Tree Diagram]

Problem 2

A. Two trees are equal if they have the same shape and the same label in corresponding nodes. Write a recursive algorithm in pseudo-code to test whether two trees are equal.

B. Using your solution to (A) as a subroutine, write an algorithm in pseudo-code to test whether tree $U$ is a subtree of $V$.

If you really want, you can answer either or both parts of this question by writing Java code; but I prefer that you get practice writing readable pseudo-code.
Problem 3

Consider the tree shown in Problem 1. Draw a picture of the data structure (objects and links) in each of the following implementations:

A. A node in the tree has a pointer to its parents but not to its children. That is, the class definition is as follows:

```java
class TreeNodeA {
    int value;
    TreeNodeA parent;
}
```

B. A node can have a maximum of 2 children. A node points to its children but not to its parent. That is, the class definition is as follows:

```java
class TreeNodeB {
    int value;
    TreeNodeB firstChild;
    TreeNodeB secondChild;
}
```

C. A node can have arbitrarily many children, arranged as a linked list. As in the illustration under "Second implementation" in the class notes, a node points to its parent, its first child, and its nextSibling. That is, the class definition is as follows:

```java
class TreeNodeC {
    int value;
    TreeNodeC parent;
    TreeNodeC firstChild;
    TreeNodeC nextSibling;
}
```

Problem 4

For each of the class definitions in Problem 3, write in Java a static method

```java
    public static TreeNode addRoot(int value, TreeNode X, TreeNode Y)
```

(where, of course, the class TreeNode should be TreeNodeA, TreeNodeB, or TreeNodeC depending). The method addRoot takes as argument two TreeNode, X and Y, that are roots of trees, and makes both X and Y children of a new node with value v. The new node is the root of a new tree, and is the value returned. Note: This is only a few lines of code, in each case.