This document provides the necessary instructions for completing the Week 9 lab exercises.

**Exercise 1: Trees**

For the tree given in the Figure 8.3, answer to the following questions:

****

1. Which node is the root?
2. What are the internal nodes?
3. How many descendants does node **cs016/ have**?
4. How many ancestors does node **cs016/ have**?
5. What are the siblings of node **homeworks**/?
6. Which nodes are in the subtree rooted at node **projects/**?
7. What is the depth of node **papers/**?
8. What is the height of the tree?

**Exercise 2: Tree traversal**

For the following trees:

1. Provide the output of the *preorder* traversal manually.
2. Draw the output of the *postorder* traversal manually.

|  |  |
| --- | --- |
|  |  |

**Exercise 3: Binary Trees**

Which of the following binary tress is proper:

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

**Exercise 4: Binary Trees – traversal**

For the following binary trees, provide the output of the *inorder* traversal (manually):

|  |  |
| --- | --- |
|  |  |

**Exercise 5: Binary Trees – Java implementation**

In the array-based implementation of a binary tree, a node v is stored in an array A at A[rank(v)], using the following algorithm:

* If v = root, rank(root) = 0
* If node is the left child of parent(node), rank(node) = 2 \* rank(parent(node)) + 1
* If node is the right child of parent(node), rank(node) = 2 \* rank(parent(node)) + 2

See the picture below and write and test a Java method to implement this algorithm.

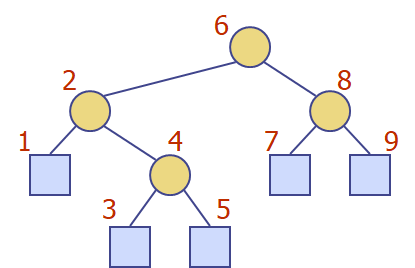
|  |  |
| --- | --- |
|  |  |

Your method should take as input a node from the above binary tree, calculate its rank and place it in the array A.

**Exercise 6: Binary Trees – Java implementation**

In the main method of LinkedBinaryTree class, implement the following:

1. Create a LinkedBinaryTree for the binary tree below:



1. Perform a preorder traversal of this linked binary tree and display the results.
2. Perform an *inorder* traversal of this linked binary tree and display the results.
3. Perform a *postorder* traversal of this linked binary tree and display the results.
4. Perform a *breadth-first* traversal and display the results.