1. Consider the parse tree for \((9 + (5 \times 3)) / (8 - 4)\):

![Parse Tree Diagram]

Indentify the following items in the above tree:

a. node containing “\(*\)”

b. edge from node containing “-” to node containing “8”

c. root node

d. children of the node containing “+”

e. parent of the node containing “3”

f. siblings of the node containing “*”

g. leaf nodes of the tree

h. subtree who’s root is node contains “+”

i. path from node containing “+” to node containing “5”

j. branch from root node to “3”

k. mark the levels of the tree (level is the number of edges on the path from the root)

l. What is the height (max. level) of the tree?

2. In Python an easy way to implement a tree is as a list of lists where a tree look like:

```
[ "node value", remaining item are subtrees for the node each implemented as a list of lists]
```

What would the list of lists representation look like for the above parse tree?
3. Consider the following Binary Tree module for the list of lists implementation:

```python
def BinaryTree(r):
    return [r, [], []]

def getRootVal(root):
    return root[0]

def setRootVal(root, newVal):
    root[0] = newVal

def getLeftChild(root):
    return root[1]

def getRightChild(root):
    return root[2]

def insertLeft(root, newBranch):
    t = root.pop(1)
    if len(t) > 1:
        root.insert(1, [newBranch, t, []])
    else:
        root.insert(1, [newBranch, [], []])
    return root

def insertRight(root, newBranch):
```

a) Complete the `insertRight` function definition.
4. Consider the following BinaryTree class for the “pointer” implementation:

```python
class BinaryTree:
    def __init__(self, rootObj):
        self.key = rootObj
        self.left = None
        self.right = None

    def getRootVal(self):
        return self.key

    def setRootVal(self, obj):
        self.key = obj

    def getLeftChild(self):
        return self.left

    def getRightChild(self):
        return self.right

    def insertLeft(self, newNode):
        if self.left == None:
            self.left = BinaryTree(newNode)
        else:
            t = BinaryTree(newNode)
            t.left = self.left
            self.left = t

    def insertRight(self, newNode):
```

a) Complete the `insertRight` method.