【K09】树的课堂练习

```
import math
nodenumber = 0

class BinaryTree:
    def __init__(self,rootObj):
        self.key = rootObj
        self.leftChild = None
        self.rightChild = None
        self.content = [rootObj,[],[]]

    def height(self):
        return int(math.log(nodenumber + 1, 2)) + 1

    def insertLeft(self,newNode):
        global nodenumber
        if self.leftChild == None:
            self.leftChild = BinaryTree(newNode)
            self.content[1] = self.leftChild.content
            nodenumber += 1
        else:
            t = BinaryTree(newNode)
            t.leftChild = self.leftChild
            self.leftChild = t
            self.content[1] = self.leftChild.content
            nodenumber += 1

    def insertRight(self,newNode):
        global nodenumber
        if self.rightChild == None:
            self.rightChild = BinaryTree(newNode)
            self.content[2] = self.rightChild.content
            nodenumber += 1
        else:
            t = BinaryTree(newNode)
            t.rightChild = self.rightChild
            self.content[2] = self.rightChild.content
            nodenumber += 1

    def getRightChild(self):
        return self.rightChild
```

```
    def getleftChild(self):
        return self.leftChild

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

    def getRootVal(self):
        return self.key

    def __str__(self):
        return str(self.content)

def buildTree():
    t = BinaryTree('a')
    t.insertLeft('b')
    t.insertRight('c')
    t.getleftChild().insertRight('d')
    t.getRightChild().insertLeft('e')
    t.getRightChild().insertRight('f')
    return t

mytree = buildTree()
print(mytree)
print(mytree.height())
```

```
['a', ['b', [], ['d', [], []]], ['c', ['e', [], []], ['f', [], []]]]
3
```