**Assignment 5**

**Level 1:  
1. (a) Define, in pseudo-code, an algorithm to calculate the height of a tree. Hint: it needs to  
be recursive.**

Algorithm treeHeight(T)

if T.isEmpty() then

return Error Message

return treeHeightHelper(T,T.root())

Algorithm treeHeightHelper(T,v)

if T.isExternal(v) then

return 0

leftHeight := treeHeightHelper(T,T.leftChild(v))

rightHeight := treeHeightHelper(T,T.rightChild(v))

if leftHeight > rightHeight then

return leftHeight +1

else

return rightHeight + 1

**(b) Using the HW05-Tree.js, implement your algorithm in a JavaScript function Height(T).**

function height(T) {

    // your code goes here Hint: you need a helper function

    if (T.isEmpty()) {

        return "Empty Tree"

    }

    return heightHelper(T, T.root());

}

function heightHelper(T, p) {

    if T.isExternal(p) then

    return 0

    let leftHeight:= heightHelper(T, T.leftChild(v))

    let rightHeight:= heightHelper(T, T.rightChild(v))

    if leftHeight > rightHeight then

    return leftHeight + 1

    else

    return rightHeight + 1

}

**(c) Implement your algorithm using the EulerTour template class provided in HW05-  
Tree.js, i.e., create a subclass of EulerTour with a method that calculates the height.**

class Height extends EulerTour {

    // add your code here, i.e., override methods from EulerTour class

    visitExternal(T, p, result) {

        result[1] = 0;

        return result[1];

    }

    visitPostOrder(T, p, result) {

        if(result[0] > result[2]){

            result[1] = result[0]+1;

        }else{

            result[1] = result[2]+1;

        }

        return result[1];

    }

    eulerTour(T, p) {

        let result = new Array(3);

        if (T.isExternal(p)) {

            this.visitExternal(T, p, result);

        } else {

            // this.visitPreOrder(T, p, result);

            result[0] = this.eulerTour(T, T.leftChild(p));

            // this.visitInOrder(T, p, result);

            result[2] = this.eulerTour(T, T.rightChild(p));

            this.visitPostOrder(T, p, result);

        }

        return result[1];

    }