# Lab 10

## CSE 274

**Note**: Please do not use `package` keyword in the `Application.java` file.
**Note**: Please do not make the `Application` class public.
**Note**: Please do not set other names for the `Application` class.

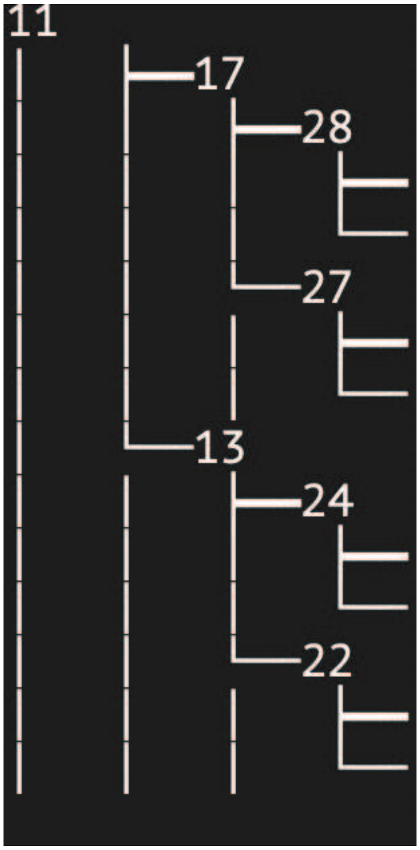
### I. A Binary Tree

Copy the following lines of code into the body of the `main` method, run the application and make sure that you see the expected output.

```
BinaryTree myBinaryTree = new BinaryTree();

myBinaryTree.root=new Node(11);
myBinaryTree.root.up= new Node(17);
myBinaryTree.root.down= new Node(13);
myBinaryTree.root.down.down= new Node(22);
myBinaryTree.root.down.up= new Node(24);
myBinaryTree.root.up.down= new Node(27);
myBinaryTree.root.up.up= new Node(28);
myBinaryTree.display();
```

The expected output is printed below:



```
11
│    ├──17
│    │    ├──28
│    │    │    ├──
│    │    │    └──
│    │    └──27
│    │    │    ├──
│    │    │    └──
│    └──13
│    │    ├──24
│    │    │    ├──
│    │    │    └──
│    │    └──22
│    │    │    ├──
│    │    │    └──
```

(A) What is the length of the above tree? (B) How many levels there exist in the tree? (C) What is the value of the root node of the tree? (D) How many children the node `17` has? (E) How many children the node `24` has? (F) Which nodes are the children of node `13`? (G) Which nodes are children of the root node? (H) Is this a binary tree? (I) What is the `up` child for node 13? (J) What is the `down` child for node 13?

## II. The NodeCounter Method

In this section, we use the `NodeCounter` method of the `BinaryTree` class to count the number of the nodes in the tree. Copy the following lines of code into the body of the `main` method, run the application and make sure that you see the expected output:

```
BinaryTree myBinaryTree = new BinaryTree();
myBinaryTree.root=new Node(11);
myBinaryTree.root.up= new Node(17);
myBinaryTree.root.down= new Node(13);
myBinaryTree.root.down.down= new Node(22);
myBinaryTree.root.down.up= new Node(24);
myBinaryTree.root.up.down= new Node(27);
myBinaryTree.root.up.up= new Node(28);
System.out.println(myBinaryTree.NodeCounter());
```

The expected output is printed below:

```
7
```

The `NodeCounter` method is as follows:

```
public int NodeCounter()
{return RecursiveNodeCounter(root);}
```

As it can be seen above, the `NodeCounter` method calls `RecursiveNodeCounter(root)` method to count the number of nodes in the tree. The `RecursiveNodeCounter` method is as follows:

```
public int RecursiveNodeCounter(Node mynode)
{
      if (mynode==null)
      return 0;

      return 1 + RecursiveNodeCounter(mynode.down)+
                 RecursiveNodeCounter(mynode.up);
}
```

The `RecursiveNodeCounter(mynode)` method counts the number of nodes in the subtree that starts at `mynode`, **recursively**.

## III.

In this section, we develop a `Length` method that calculates the length of the binary tree:

```
public int Length()
{return RecursiveLength(root);}
```

As it can be seen above, the `Length` method calls the `RecursiveLength(root)` method to obtain the length of the tree. Develop the `RecursiveLength` method for the `BinaryTree` class:

```
 public int RecursiveLength(Node mynode)
```

The above method takes `mynode` as an input and calculates the length of the subtree that starts at `mynode`, **recursively**. Use the following lines of code to test the developed method:

```
 BinaryTree myBinaryTree = new BinaryTree();

myBinaryTree.root=new Node(11);
myBinaryTree.root.up= new Node(17);
myBinaryTree.root.down= new Node(13);
myBinaryTree.root.down.down= new Node(22);
myBinaryTree.root.down.up= new Node(24);
myBinaryTree.root.up.down= new Node(27);
myBinaryTree.root.up.up= new Node(28);
System.out.println(myBinaryTree.Length());
```

The expected output is printed below:

```
 3
```

## IV. Submitting the Assignment

When submitting your response to this assignment, keep the above lines of code in the body of the `main` method.