**Height of a Binary Tree**

For a tree with just one node, the root node, the height is defined to be 0, if there are 2 levels of nodes the height is 1 and so on. A null tree (no nodes except the null node) is defined to have a height of $-1$.

The following `height` function in pseudocode is defined recursively as discussed in class. It should be easily to add a C++ version to the program `binTree1.cpp`.

```plaintext
int height( BinaryTree Node t) {
    if t is a null tree
        return -1;
    hl = height( left subtree of t);
    hr = height( right subtree of t);
    h = 1 + maximum of hl and hr;

    return h;
}
```

For example, the following tree has a height of 4. Its left subtree has height 2 and its right subtree 3.

---

For example, the following tree has a height of 4. Its left subtree has height 2 and its right subtree 3.
**Exercise 1**

Add a method called `height()` to the program `binTree1.cpp` for calculating tree heights. Modify the `main()` function so that `height()` is called.

**Exercise 2**

Modify the `main()` function so that it creates the following binary tree:

```
13
  /   \
10     15
 /     /
1   12  13
/     /   \
11   17   21
     /     \
     18
```

Then run the traversal and height methods on it. See how they behave by checking the order of traversal against this tree. What height is returned for it?