Trees

A (rooted) tree is defined recursively:

- A tree is either:
  - empty
  - a root node with one or more children, each of which is a tree.

Tree Terminology

- node
- edge
- root node
- internal nodes (nodes with children)
- external nodes / leaves

More Tree Terminology

- c is a parent of e, f, and g. It is an ancestor of e, f, g, h, and i.
- h is a child of e; it is a descendant of e, c, and a.
- e, f, and g are siblings
- The circled nodes form a subtree rooted at b.

More Tree Terminology

- The depth of a node is the number of edges between it and the root node.
- The height of a tree is the maximum depth of any node; this tree has height 3.

Binary Trees

A binary tree is defined recursively:

- A tree is either:
  - empty
  - a root node with a left child and a right child, each of which is a binary tree.
Binary Trees

Yet More Terminology

A tree is a full tree if:
* Every leaf node is at the same depth, and
* Every internal node has two children

Implementing the Binary Tree

Just follow the recursive definition to get a simple implementation:

```cpp
template <class T>
class binary_tree_node {
public:
    T data;
    binary_tree_node<T>* left;
    binary_tree_node<T>* right;
};
```
Implementing the Binary Tree

• For now, we’ll just implement a tree as nodes
• Tree functions will be global
• Can also wrapper specific kinds of binary trees as classes

Binary Tree Traversals

• A traversal of a tree is the act of visiting every node in the tree once.
• There are three traversal orders:
  – Pre-order
  – In-order
  – Post-order

Pre-Order Traversal

Visit the root first, then the left and right sub-trees:

```
  1
 / \     \
 2   7   \
 / \     \
3   4   8
 / \     \
5   6
```

In-Order Traversal

Visit the left sub-tree, the root, and then the right sub-tree:

```
  2
 / \     \
 6   8   \
 /     / \
1     3   5
```

Post-Order Traversal

Visit the left and right sub-trees first and the root last:

```
  8
 / \     \
 5   7   \
 /     / \
1     4   6
 /     / \
2     3
```

Pre-Order Traversal Implementation

Note naturally recursive description: visit the root first, then the left and right sub-trees.

So we get a naturally recursive implementation:
```
template <class T>
void do_preorder(binary_tree_node<T>* root) {
    if (root != NULL) {
        // do something with root->data
        do_preorder(root->left);
        do_preorder(root->right);
    }
}
```
Other Implementations

Can you write the in-order and post-order traversal code?