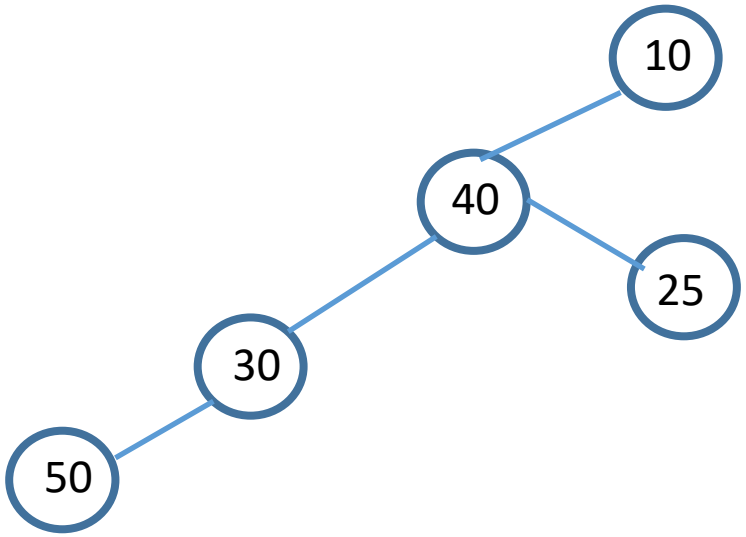
COMP 2611, Data Structures

LECTURE 7: BINARY TREES AND BINARY SEARCH TREES

Give the Preorder, Inorder, and Postorder Traversals of This Binary Tree:



Preorder: 10 40 30 50 25

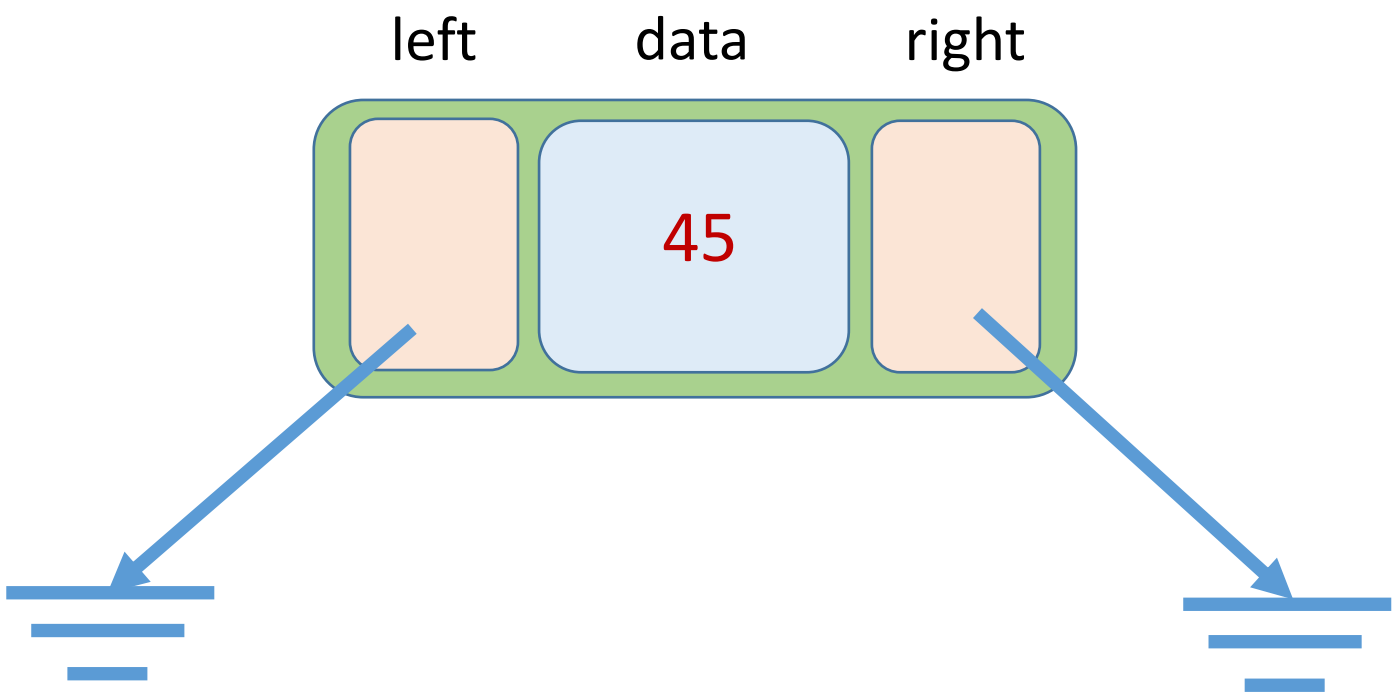
Inorder: 50 30 40 25 10

Postorder: 50 30 25 40 10

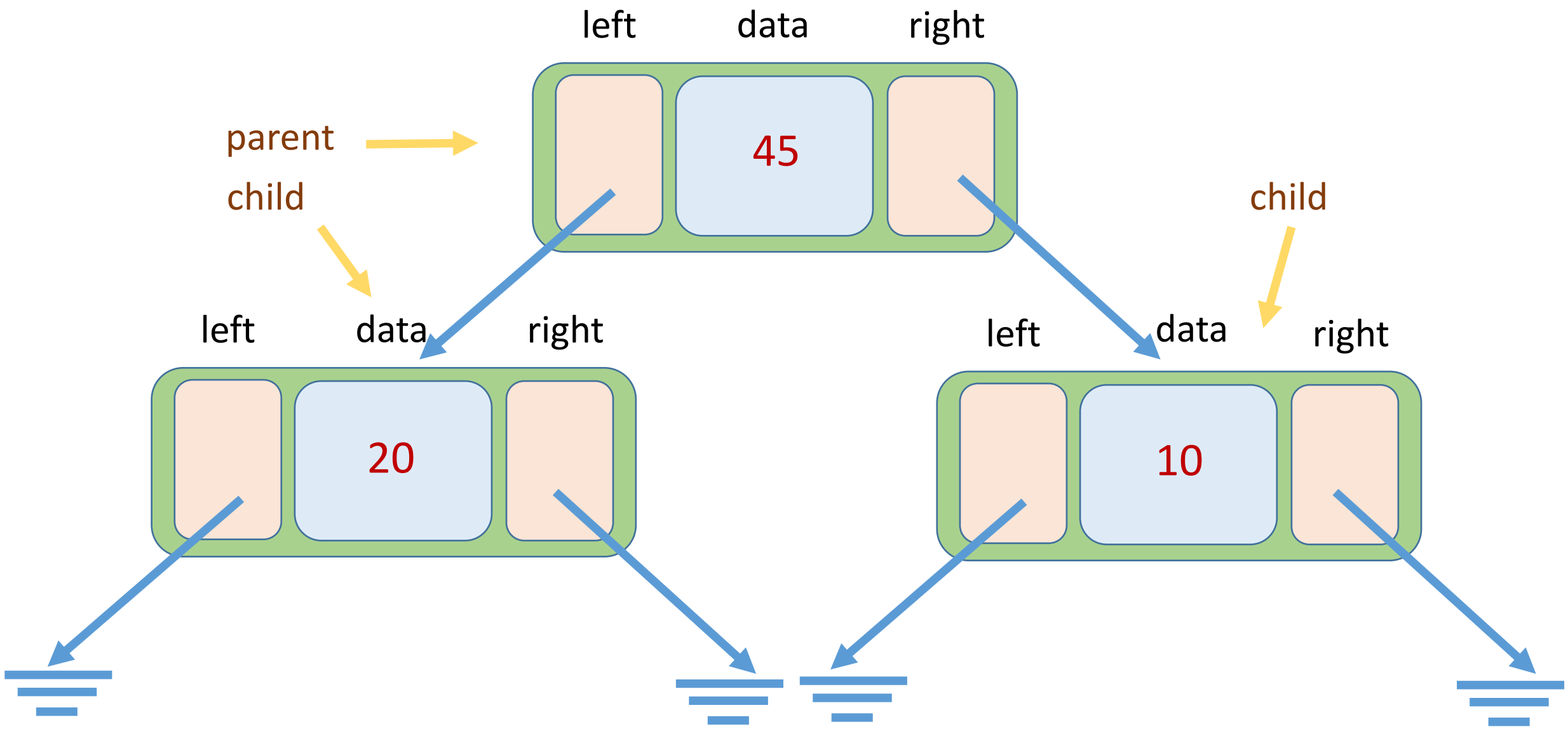
Implementation in C++

We will now discuss how to implement a binary tree and its related algorithms in C++.

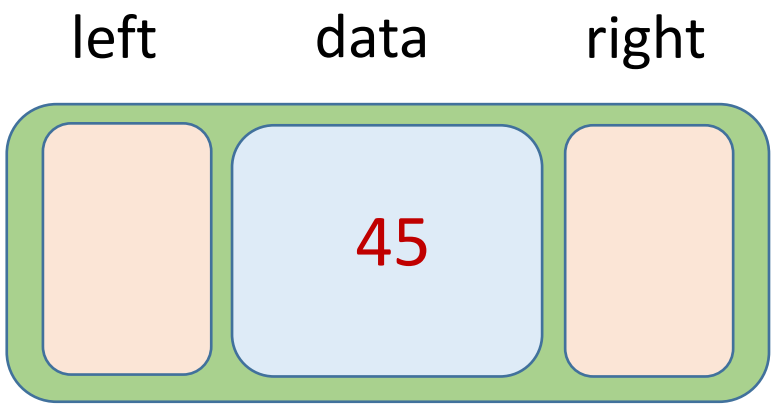
A Node in a Binary Tree



Implementation of Nodes in a Binary Tree

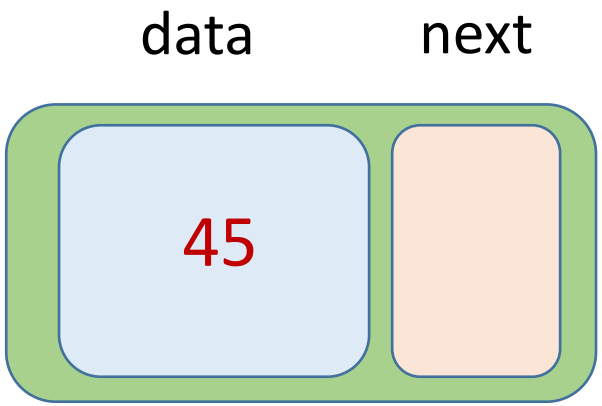


Declaring a Node in a Binary Tree




```
struct BTNode {
      int data;
      BTNode * left;
      BTNode * right;
};
```

Declaring a Node in a Linked List




```
struct LLNode {
      int data;
      LLNode * next;
};
```

Exercise

Wrte the code for the *preOrder*, *inOrder*, and *postOrder* functions with the following prototypes:

```
void preOrder (BTNode * root);
void inOrder (BTNode * root);
void postOrder (BTNode * root);
```

The functions must all be recursive and should simply display the value stored in the node when it is “visited”.

Solution for Exercise (preorder)

```
void preOrder (BTNode * root) {

    if (root == NULL)
      return;

    cout << root->data << endl;

    preOrder (root->left);
    preOrder (root->right);

}
```

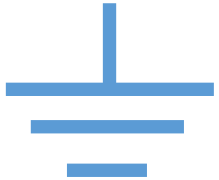
Exercise

Write a recursive function, *numNonTerminal*, with the following prototype, which returns the number of non-terminal nodes in a binary tree:

```
int numNonTerminal (BTNode * root);
```

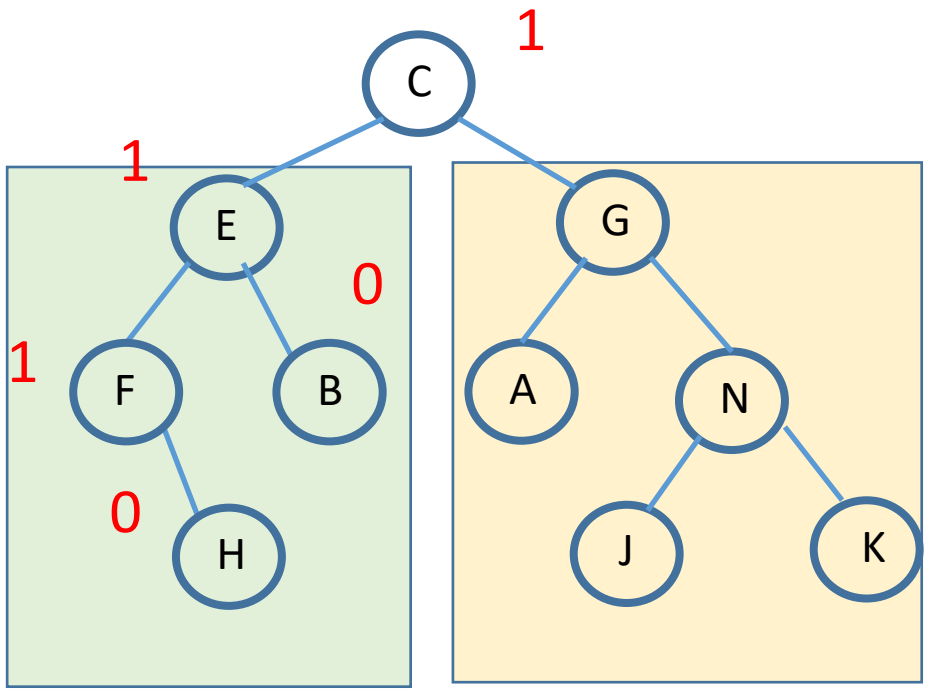
A non-terminal node is any node that is not a leaf.

Cases to Consider



(Empty tree)

(Tree has one node)



(Tree has non-empty left or right subtrees)

Implementing the Cases

The binary tree is empty: return 0

The root of the binary tree has no children : return 0

The root of the binary tree has 1 or 2 children: add 1 to count of non-terminal nodes in the rest of the tree

The number of non-terminal nodes in the rest of tree =

the number on the left-side of the root + the number on the right-side of the root

```
numNonTerminal (root->left)   +   numNonTerminal (root->right)
```

Code for Function

```
int numNonTerminal (BTNode * root) {

   if (root == NULL)
      return 0;        // tree is empty

   if (root->left == NULL && root->right == NULL)
      return 0;        // root has no children

   return (1 + numNonTerminal (root->left) + numNonTerminal (root->right));
                       // add 1 to left-side count + right-side count
}
```

Call *numNonTerminal* With This Binary Tree: