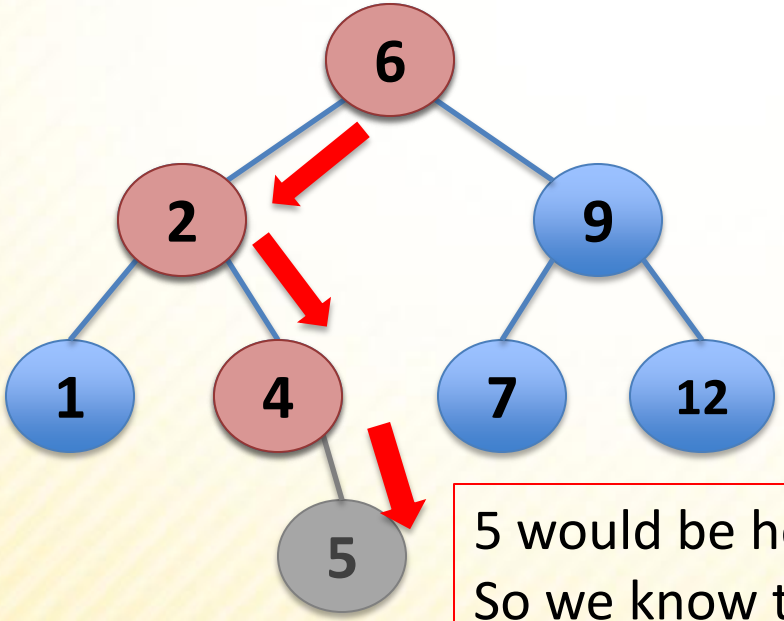
Binary Trees & Insertion

COP 3502

Binary Search Tree Insertion

- Inserting a Node into a Binary Search Tree
  - Similar to searching for a node
  - We have to “trace out” the same path, to find where this node belongs in the tree.
  - Let’s say we were going to search for 5 in the following tree:



5 would be here if it was in the tree, So we know this is where 5 belongs when inserting the value.

Binary Search Tree Insertion

- Strategy to insert a node recursively:
  1) If the tree is empty, just return a pointer to a node containing the new value.
  2) Otherwise see which sub-tree the node should be inserted into by comparing the value stored at the root.
     a) If we want to insert in the left sub-tree and it's NULL, then we know this is where we attach the node so no recursive call is necessary.
     b) Same with the right…
  3) Then recursively either insert into the left sub-tree or the right sub-tree.

Binary Search Tree Insertion

- Create the code in class.

```
node* insertNode(node *curr, node *temp) {
    // Inserting into an empty tree.
    // temp2 should be inserted to the right.
    // There is a right subtree to insert the
    // Place the node directly to the right of
    // temp2 should be inserted to the left.
    // There is a left subtree to insert the node.
    // Place the node directly to the left of
    // Return the curr pointer of the updated tree.
}
```

Binary Search Tree Insertion

- Run some insertion, traversal, and search examples in code.
- Draw out some examples on the board.

Summing the nodes in a Binary Tree

- We can really use any of the traversals to implement this.
  - All we need to do add the values from the three portions of the three together and return this answer.
    - Notice how succinct this code is!

```
int Add(struct tree_node *current_ptr) {

     if (current_ptr != NULL)
          return current_ptr->data+
                 Add(current_ptr->left)+
                 Add(current_prt->right);
     else
          return 0;
}
```