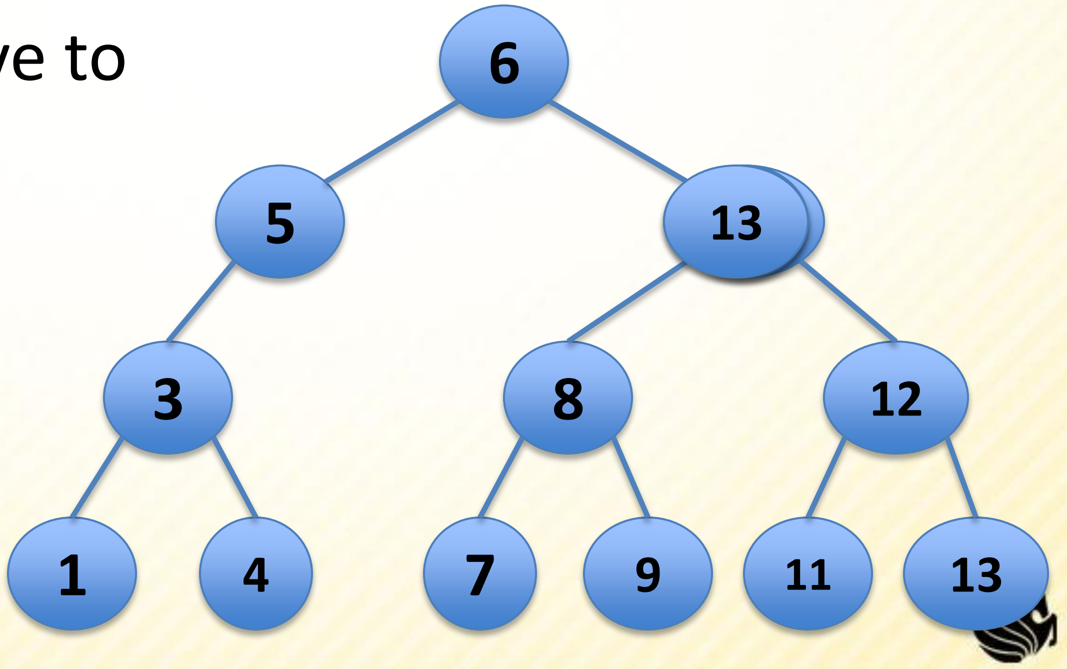
BINARY SEARCH TREES DELETION

COP 3502

Binary Search Tree - Deletion

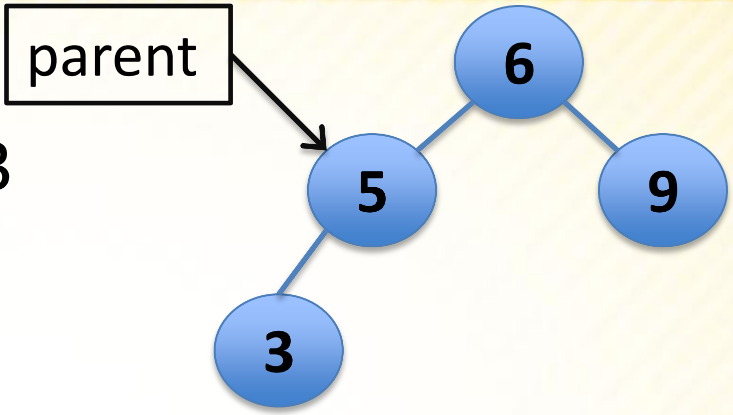
- Let’s consider several cases for Deletion:
  1) Deleting a Leaf node
  2) Deleting a node with 1 child
  3) Deleting a node with 2 children
  - For all cases we have to identify the parent.



Binary Search Tree - Deletion

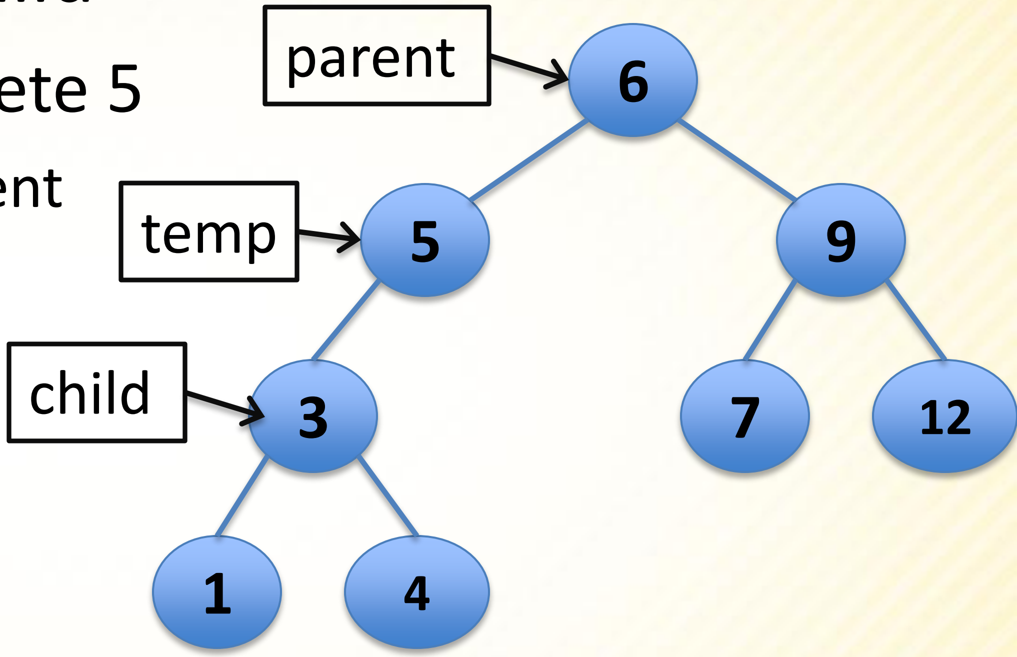
- Deleting a leaf node
  - Let's say we want to delete 3
    - Find the parent of 3
      - which is 5
    - Then what?
    - free(parent->left);
    - parent->left = NULL;

    - or if it's a right leaf node:
    - free(parent->right);
    - parent->right= NULL;



Binary Search Tree - Deletion

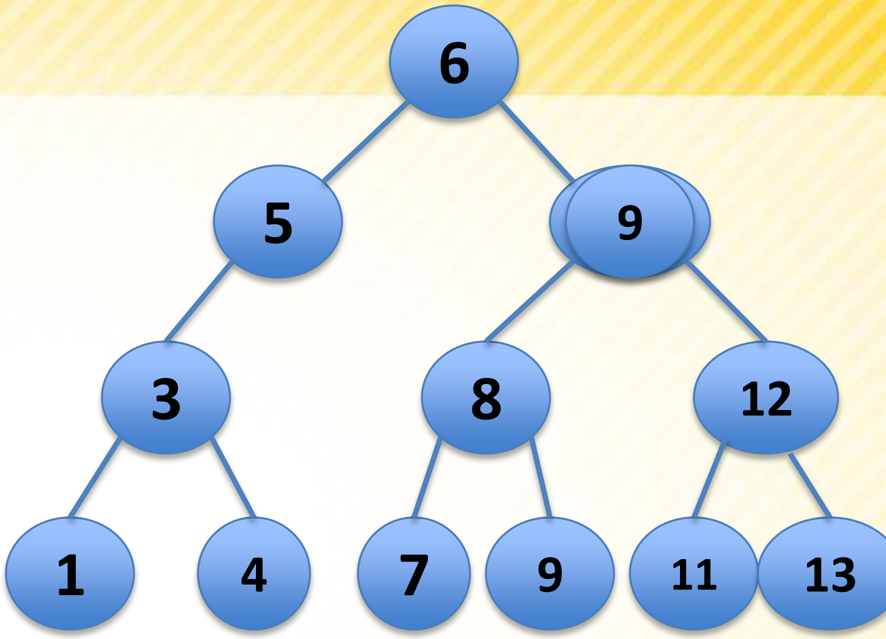
- Deleting node with 1 child
  - Let's say we want to delete 5
    - We need to find the parent
    - AND the sole child
    - Then connect them



    - temp = parent->left;
    - child = temp->left;
    - parent->left = child;
    - free(temp);

Binary Search Tree - Deletion

- Deleting a node with 2 children:
  - In order to maintain the **BST property**, we can replace the deleted node with either:
    - ➢Max in the left subtree
    - ➢OR Min in the right subtree



Binary Search Tree Deletion

- In aiding in Deletion, we will want several auxiliary functions:
  1) **node* parent(node* root)**
     - Finds the parent of a given node in a binary tree.
  2) **node* minVal(node* root), node* maxVal(node* root)**
     - Finds the minimum (or maximum) value in a given binary tree.
  3) **int isLeaf(node* root)**
     - Determines if a node is a leaf node or not.
  4) **int hasOnlyLeftChild(node* root),**
     int hasOnlyRightChild(node* root)
     - Determines if a node ONLY has a left (or right) child or not.
  5) **node* findNode(node* root)**
     - Returns a pointer to a node in a given tree that stores a particular value.

```
node* delete(node* root, int value) {
  node *delnode, *parent;

  // Get a pointer to the node to delete.
  delnode = findNode(root, value);

  // Get the parent of this node.
  parent = parent(root, delnode);

  // Case 1:  the node to delete is a leaf node.
  if (isLeaf(delnode)) {...}

  // Case 2: node to be deleted only has a left child.
  if (hasOnlyLeftChild(delnode)) {...}

  // Case 3: node to be deleted only has a right child.
  if (hasOnlyRightChild(delnode)) {...}

  // Case 4:  the node has 2 children
  else {...}
}
```

Binary Search Tree Deletion

- Fill in the code together in class for the Delete function.