Binary Search Tree - Delete

```
typedef struct {
   int data;
   node *left;
   node *right;
} node;

// Deletes value from a BST rooted at root. value must be in the tree in
// to work.  Returns a pointer to the root of the resulting tree.
node* delete(node* root, int value) {
  node *delnode, *new_del_node, *save_node, *par;
  int save_val;

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

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

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

    // Deleting the only node in the tree.
    if (par == NULL) {
      free(delnode); // free the memory for the node.
      return NULL;
    }

    // Deletes the node if it's a left child.
    if ( value < par->data ) {
      free( delnode ); // Free the memory for the node.
      par->left = NULL;
    }

    // Deletes the node if it's a right child.
    else {
      free( delnode ); // Free the memory for the node.
      par->right = NULL;
    }

    return root; // Return the root of the new tree.
  }

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

    // Deleting the root node of the tree.
    if ( par == NULL ) {
      save_node = delnode->left;
      free( delnode ); // Free the node to delete.
      return save_node // Return the new root node of the resulting tree.
    }

    // Deletes the node if it's a left child.
```