**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 = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_; // Get a pointer to the node to delete.**

**par = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_; // Get the parent of this node.**

**// Case 1: the node to delete is a leaf node.**

**if (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_) {**

**// Deleting the only node in the tree.**

**if (par == NULL) {**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_; // free the memory for the node.**

**return \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;**

**}**

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

**if (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_) {**

**free(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_); // Free the memory for the node.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ = NULL;**

**}**

**// Deletes the node if it's a right child.**

**else {**

**free(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_); // Free the memory for the node.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ = NULL;**

**}**

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

**}**

**// Case 2: the node to be deleted only has a left child.**

**if (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_) {**

**// Deleting the root node of the tree.**

**if (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_) {**

**save\_node = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;**

**free(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_); // Free the node to delete.**

**return \_\_\_\_\_\_\_\_\_; // Return the new root node of the resulting tree.**

**}**

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

**if (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_) {**

**save\_node = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_; // Save the node to delete.**

**par->left = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_; // Readjust the parent pointer.**

**free(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_); // Free the memory for the deleted node.**

**}**

**// Deletes the node if it's a right child.**

**else {**

**save\_node = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_; // Save the node to delete.**

**par->right = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_; // Readjust the parent pointer.**

**free(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_); // Free the memory for the deleted node.**

**}**

**return root; // Return the root of the tree after the deletion.**

**}**

**// Case 3: the node to be deleted only has a right child.**

**if (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_) {**

**// Node to delete is the root node.**

**if (par == NULL) {**

**save\_node = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;**

**free(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_);**

**return \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;**

**}**

**// Delete's the node if it is a left child.**

**if (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_) {**

**save\_node = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;**

**par->left = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;**

**free(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_);**

**}**

**// Delete's the node if it is a right child.**

**else {**

**save\_node = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;**

**par->right = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;**

**free(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_);**

**}**

**return root;**

**}**

**// Case 4: The deleted node has 2 children, find the replacement node**

**new\_del\_node = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;**

**save\_val = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;**

**delete(root, \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_); // Now, delete the proper value.**

**// Restore the data to the original node to be deleted.**

**delnode->data = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;**

**return root;**

**}**