Deletion in a Binary Tree

Problem: Given a Binary Tree and a node to be deleted from this tree. The task is to delete the given node from it.

Examples:

```
Delete 10 in below tree
                                                    10
                                                  /    \
                                                 20     30

Output :
                                                    30
                                                  /
                                                 20


Delete 20 in below tree
                                                   10
                                                  /    \
                                                 20     30
                                                         \
                                                          40
Output :
                                                    10
                                                  /   \
                                                 40    30
```

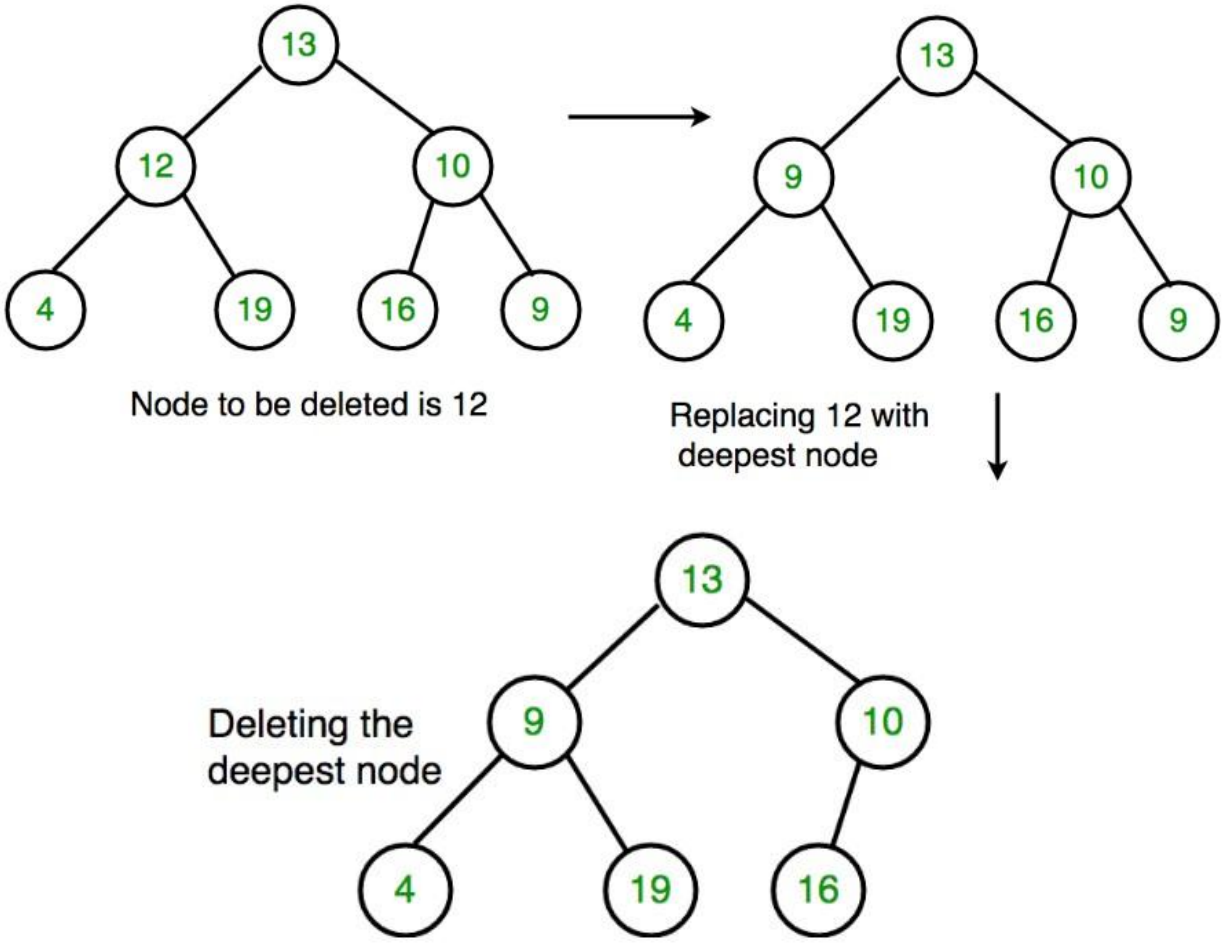
While performing the delete operation on binary trees, there arise a few cases:

1. The node to be deleted is a leaf node. That is it does not have any children.
2. The node to be deleted is a internal node. That is it have left or right child.
3. The node to be deleted is the root node.

In the first case 1, since the node to be deleted is a leaf node, we can simply delete the node without any overheads. But in the next 2 cases, we will have to take care of the children of the node to be deleted.

In order to handle all of the cases, one way to delete a node is to:

1. Starting at the root, find the deepest and rightmost node in binary tree and node which we want to delete.
2. Replace the deepest rightmost node's data with the node to be deleted.
3. Then delete the deepest rightmost node.


13
12
10
4
19
16
9
Node to be deleted is 12
13
9
10
4
19
16
9
Replacing 12 with
deepest node
Deleting the
deepest node
13
9
10
4
19
16

Below is the implementation of the above approach:

C++ Java

```cpp
// C++ program to delete element in binary tree
#include <bits/stdc++.h>
using namespace std;
// Binary Tree Node
struct Node
{
    int key;
    struct Node* left, *right;
};
// Utility function to create a new Binary Tree Node
struct Node* newNode(int key)
{
    struct Node* temp = new Node;
    temp->key = key;
    temp->left = temp->right = NULL;
    return temp;
};
// Function to perform Inorder Traversal
void inorder(struct Node* temp)
{
    if (!temp)
        return;
    inorder(temp->left);
    cout << temp->key << " ";
    inorder(temp->right);
}
// Function to delete the given deepest node (d_node) in binary tree
void deletDeepest(struct Node *root, struct Node *d_node)
{
    queue<struct Node*> q;
    q.push(root);

    // Do level order traversal until last node
    struct Node* temp;
    while(!q.empty())
    {
        temp = q.front();
        q.pop();

        if (temp->right)
        {
            if (temp->right == d_node)
            {
                temp->right = NULL;
                delete(d_node);
                return;
            }
            else
                q.push(temp->right);
        }
```

```
        if (temp->left)
        {
            if (temp->left == d_node)
            {
                temp->left=NULL;
                delete(d_node);
                return;
            }
            else
                q.push(temp->left);
        }
    }
}
// Function to delete element in binary tree
void deletion(struct Node* root, int key)
{
    queue<struct Node*> q;
    q.push(root);
    struct Node *temp;
    struct Node *key_node = NULL;
    // Do level order traversal to find deepest
    // node(temp) and node to be deleted (key_node)
    while (!q.empty())
    {
        temp = q.front();
        q.pop();

        if (temp->key == key)
             key_node = temp;

         if (temp->left)
             q.push(temp->left);
         if (temp->right)
             q.push(temp->right);
    }
    int x = temp->key;
    deletDeepest(root, temp);
    key_node->key = x;
}
// Driver code
int main()
{
    // Create the following Binary Tree
    //            10
    //         /      \
    //        11       9
    //       /  \     / \
    //      7    12  15  8
    struct Node* root = newNode(10);
    root->left = newNode(11);
    root->left->left = newNode(7);
    root->left->right = newNode(12);
    root->right = newNode(9);
    root->right->left = newNode(15);
    root->right->right = newNode(8);
    cout << "Inorder traversal before deletion : ";
    inorder(root);
```

```
    int key = 11;
    deletion(root, key);
    cout << endl;
    cout << "Inorder traversal after deletion : ";
    inorder(root);
    return 0;
}
```

Output:

```
Inorder traversal before Deletion: 7 11 12 10 15 9 8
Inorder traversal after Deletion: 7 8 12 10 15 9
```