1.

We use a while loop to start the search, and if-else statements to test if we should go left or right. If we could not find it, return false and exit the while loop, meaning that the input node values is not in the tree.

If we can find the value inside, we can start the deletion. We first use an if statement to test when the node has no left children nor right children, so simply delete the node, making it null without needing to reconnect the children after it. The first if statement tests if the node is the root of the tree, then because it has no right or left children, the tree is empty. The following else-if and else statements test whether it is the left or right children of the root and simply delete the child.

In the cases where there is only one child, we can delete the Node cur and make the parent of it as the parent of the target Node. The next else-if statement test if there is no right child of the node, replace the rest with the left subtree. The first if statement tests if the current node is the root, then simply replace the root with the left child. The following else-if and else statement test if the key node is the left or right child, and then replace the left or right child with the left child.

Similar to the above else-if case where there is no right child, when there is no left child, simply replace it with the right subtree. The first if statement tests if the current node is the root, then simply replace the root with the right child. The following else-if and else statement test if the key node is the left or right child, and then replace the left or right child with the right child.

The most complicated case comes when it has both left and right child, therefore we need to replace it with inorder successor. Here is why we need it and how it works. When a node with children is deleted, it will not automatically convert to the proper form to connect, therefore, we need to use a getSuccessor method to connect it back properly to the tree. The getSuccessor method is used to return the node with the next highest value after the delNode and continues to go to the right child and the right child’s left descendants.

First, we need to create a new node successor and get the successor of the node to delete. Next, we need to connect the parent of the current to the successor instead. The first if statement tests if the current node is the root, if so, simply make the root the successor. The following else-if and else statement test if the left child or the right child is the successor, and connect the parent of the current to the successor. Next, connect the successor to the currents left child. Note that the successor cannot have a left child. Return true at the end of the deletion.

In this case of deletion, we can go from left of the subtree and find the largest node through traversing the right branch, or go from the right and find the smallest node traversing the left branch. Take the left first appraoch, we can then find out the number of child the target node has. If there is no child, we set the right child of parent of the target node to null, set the right child of this node to the right child of Node cur, and left child of this node to the left node of Node cur accordingly. Or we can go from the right, and delete the node accordingly just like from the left. After finding the target node, we can set the parent of Node cur to the parent of thr target Node to connect it back to the tree. In this way, we successfully deleted the node.

2. Deleting node 8

Node 8 has two children, so the first two cases having one or no children will not be demonstrated. There are two ways to delete node 8. We can go either from the left or the right side of the tree, searching for the largest or smallest node.

If we go from the left, we search for the largest node in the left subtree by traversing the right branch, which gives us 7 at the end. Because 7 has no right or left children, we set its right child of parent to null, which is Node 6 in this case. We then set the left child of node 7 to 2, and the right child to 11. Pushing up the tree, we need to set the left child of Node 15 to Node 7, and now all the nodes reconnect after the deletion of 8.

If we go from the right, we search for the smallest node in the left subtree by traversing the left branch to the end, which gives us 10. Because 10 is at the end of the tree and has no left nor right children, we set the left child of parent of node 10 to null, which is 11. We then set the left child of 10 to 2 and the right child of 10 to 11. We need to set the left child of 15 the root to 10, and therefore all the nodes connect properly after deleting 8.