```java
// Function to delete a node from BST.
// Return the root of the BST after deleting the node with value X.
public static Node deleteNode(Node root, int X) {
    if (root == null) {
        return null;
    }
    if (root.data == X) {
        // 1. If left subtree empty, make right node as new Root.
        if (root.left == null) {
            root = root.right;
            return root;
        }
        // 2. If right subtree empty, make left node as new Root.
        if (root.right == null) {
            root = root.left;
            return root;
        }
        // 3. If both subtree are present, then new root will be the largest node of left subtree.
        // 3. i) If the largest node is left node then make it root and set new root's
        // right subtree to previous root's right subtree.
        if (root.left.right == null) {
            root.left.right = root.right;
            return root.left;
        }
```

```
            // 3. ii) Find largest node, then make it root and set
            // new root's left subtree to previous root's left subtree and
            // new root's right subtree to previous root's right subtree
            Node prev = root;
            Node curr = root.left;
            Node nextRoot = curr;
            while (curr.right != null) {
                prev = curr;
                curr = curr.right;
                nextRoot = curr;
            }
            prev.right = nextRoot.left;
            nextRoot.right = root.right;
            nextRoot.left = root.left;
            return nextRoot;
        }
        // IF data of root is greater than deleting value, try to delete from left
        // subtree.
        if (root.data > X) {
            Node left = deleteNode(root.left, X);
            root.left = left;
        }
        // IF data of root is less than deleting value, try to delete from right
        // subtree.
        else {
            Node right = deleteNode(root.right, X);
            root.right = right;
        }
        return root;
    }
```