








Delete a Node with One Child in a Binary Search Tree

Now that we can delete leaf nodes let's move on to the second case: deleting a node with one child. For this case, say we have a tree with the following nodes 1 – 2 – 3 where 1 is the root. To delete 2, we simply need to make the right reference in 1 point to 3. More generally to delete a node with only one child, we make that node's parent reference the next node in the tree.

We've provided some code in our `remove` method that accomplishes the tasks from the last challenge. We find the target to delete and its parent and define the number of children the target node has. Let's add the next case here for target nodes with only one child. Here, we'll have to determine if the single child is a left or right branch in the tree and then set the correct reference in the parent to point to this node. In addition, let's account for the case where the target is the root node (this means the parent node will be `null`). Feel free to replace all the starter code with your own as long as it passes the tests.

	The <code>BinarySearchTree</code> data structure should exist.
	The binary search tree should have a method called <code>remove</code> .
	Trying to remove an element that does not exist should return <code>null</code> .
	If the root has no children, deleting it should set the root to <code>null</code> .
	The <code>remove</code> method should remove leaf nodes from the tree.
	The <code>remove</code> method should remove nodes with one child.
	Removing the root in a tree with two nodes should set the second to be the root.