

Our Solution(s)	Run Code	Your Solutions	Run Code
<div>Solution 1</div> <pre>1 // Copyright © 2020 AlgoExpert, LLC. All rights reserved. 2 3 package main 4 5 type AncestralTree struct { 6 Name string 7 Ancestor *AncestralTree 8 } 9 10 // O(d) time O(1) space - where d is the depth (height) of the tree 11 func GetYoungestCommonAncestor(topAncestor, descendantOne, descendantTwo *AncestralTree) *AncestralTree { 12 depthOne := getDescendantDepth(descendantOne, topAncestor) 13 depthTwo := getDescendantDepth(descendantTwo, topAncestor) 14 if depthOne > depthTwo { 15 return backtrackAncestralTree(descendantOne, descendantTwo, topAncestor) 16 } 17 return backtrackAncestralTree(descendantTwo, descendantOne, topAncestor) 18 } 19 20 func getDescendantDepth(descendant, topAncestor *AncestralTree) int { 21 depth := 0 22 for descendant != topAncestor { 23 depth++ 24 descendant = descendant.Ancestor 25 } 26 return depth 27 } 28 29 func backtrackAncestralTree(lowerDescendant, higherDescendant *AncestralTree) *AncestralTree { 30 for diff > 0 { 31 lowerDescendant = lowerDescendant.Ancestor 32 diff-- 33 } 34 for lowerDescendant != higherDescendant { 35 lowerDescendant = lowerDescendant.Ancestor 36 higherDescendant = higherDescendant.Ancestor 37 } 38 return lowerDescendant 39 } 40</pre>		<div>Solution 1 Solution 2 Solution 3</div> <pre>1 package main 2 3 type AncestralTree struct { 4 Name string 5 Ancestor *AncestralTree 6 } 7 8 func GetYoungestCommonAncestor(top, descendantOne, descendantTwo *AncestralTree) *AncestralTree { 9 // Write your code here. 10 return nil 11 } 12</pre>	
		<div>Custom Output</div> <div>Submit Code</div>	

Run or submit code when you're ready.