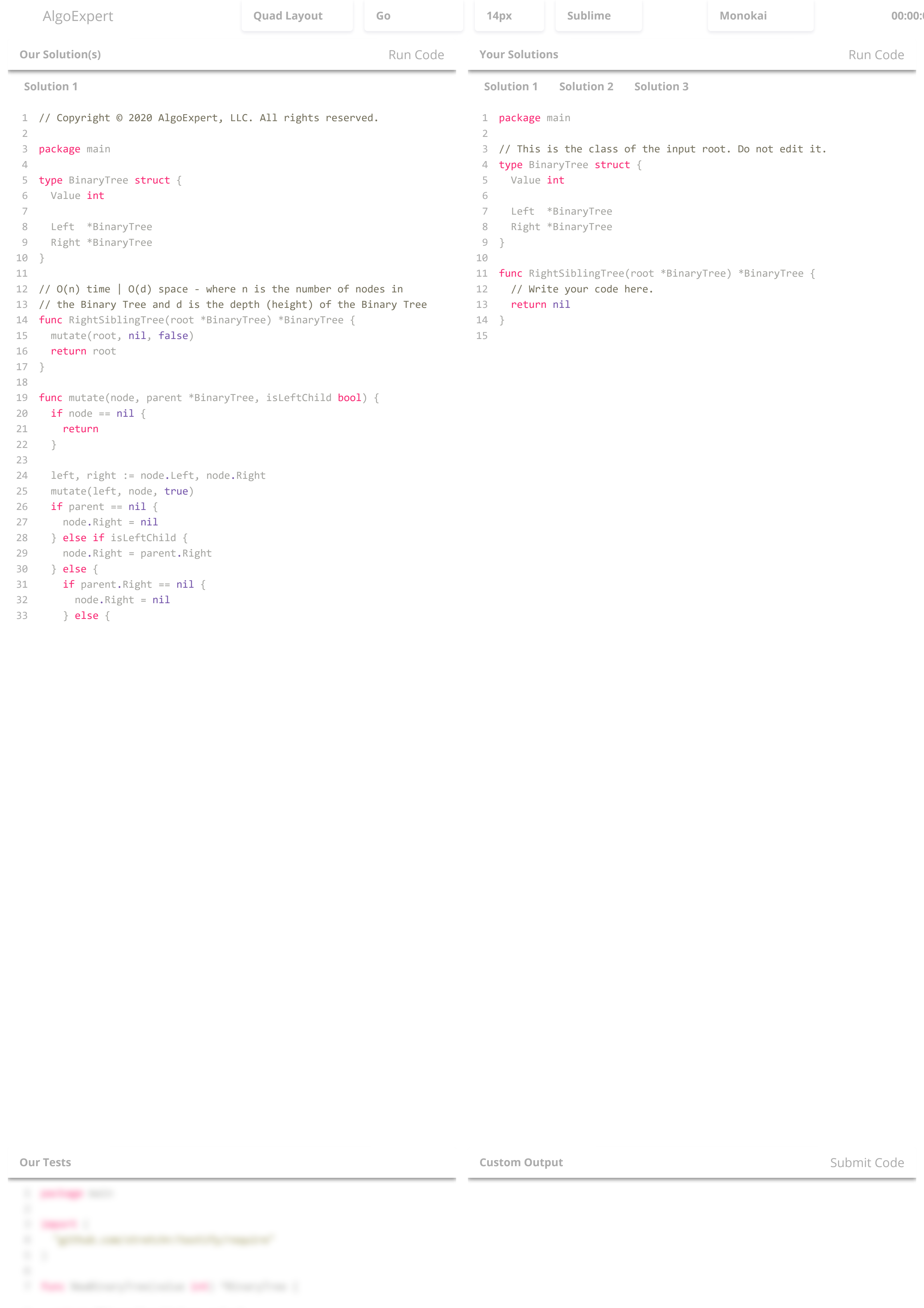

```go
// Copyright © 2020 AlgoExpert, LLC. All rights reserved.

package main

type BinaryTree struct {
  Value int

  Left  *BinaryTree
  Right *BinaryTree
}

// O(n) time | O(d) space - where n is the number of nodes in
// the Binary Tree and d is the depth (height) of the Binary Tree
func RightSiblingTree(root *BinaryTree) *BinaryTree {
  mutate(root, nil, false)
  return root
}

func mutate(node, parent *BinaryTree, isLeftChild bool) {
  if node == nil {
    return
  }

  left, right := node.Left, node.Right
  mutate(left, node, true)
  if parent == nil {
    node.Right = nil
  } else if isLeftChild {
    node.Right = parent.Right
  } else {
    if parent.Right == nil {
      node.Right = nil
    } else {
```

```go
package main

// This is the class of the input root. Do not edit it.
type BinaryTree struct {
  Value int

  Left  *BinaryTree
  Right *BinaryTree
}

func RightSiblingTree(root *BinaryTree) *BinaryTree {
  // Write your code here.
  return nil
}
```

```
  return &BinaryTree{Value: value}
}

func (tree *BinaryTree) Insert(value int) *BinaryTree {
  queue := []*BinaryTree{tree}
  for len(queue) > 0 {
    var current *BinaryTree
    queue, current = queue[1:], queue[0]
    if current.Left == nil {
      current.Left = NewBinaryTree(value)
      return tree
    }
    queue = append(queue, current.Left)

    if current.Right == nil {
```

Run or submit code when you're ready.