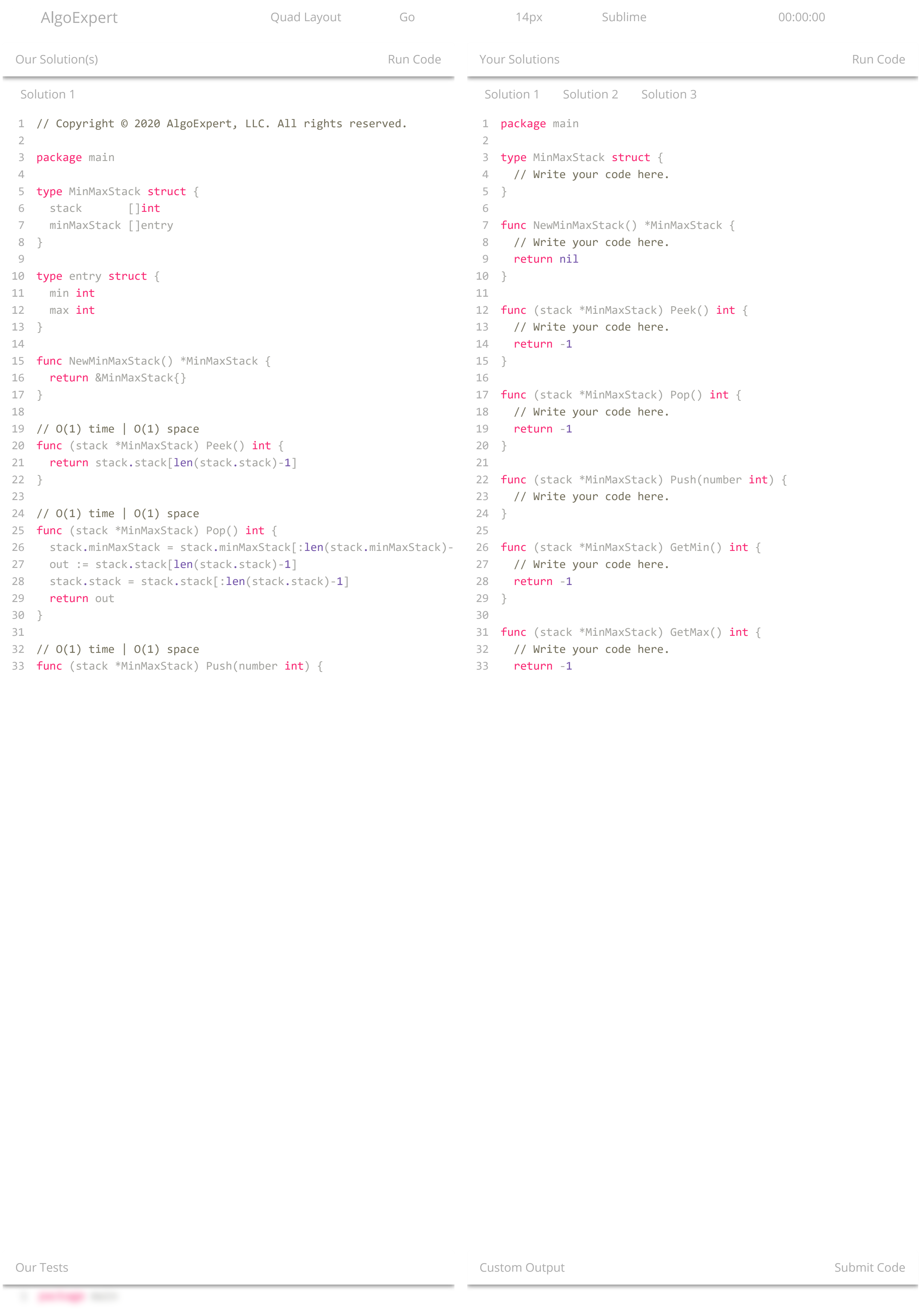

```go
// Copyright © 2020 AlgoExpert, LLC. All rights reserved.

package main

type MinMaxStack struct {
  stack       []int
  minMaxStack []entry
}

type entry struct {
  min int
  max int
}

func NewMinMaxStack() *MinMaxStack {
  return &MinMaxStack{}
}

// O(1) time | O(1) space
func (stack *MinMaxStack) Peek() int {
  return stack.stack[len(stack.stack)-1]
}

// O(1) time | O(1) space
func (stack *MinMaxStack) Pop() int {
  stack.minMaxStack = stack.minMaxStack[:len(stack.minMaxStack)-
  out := stack.stack[len(stack.stack)-1]
  stack.stack = stack.stack[:len(stack.stack)-1]
  return out
}

// O(1) time | O(1) space
func (stack *MinMaxStack) Push(number int) {
```

```go
package main

type MinMaxStack struct {
  // Write your code here.
}

func NewMinMaxStack() *MinMaxStack {
  // Write your code here.
  return nil
}

func (stack *MinMaxStack) Peek() int {
  // Write your code here.
  return -1
}

func (stack *MinMaxStack) Pop() int {
  // Write your code here.
  return -1
}

func (stack *MinMaxStack) Push(number int) {
  // Write your code here.
}

func (stack *MinMaxStack) GetMin() int {
  // Write your code here.
  return -1
}

func (stack *MinMaxStack) GetMax() int {
  // Write your code here.
  return -1
```

```go
package main
```

```
func testMinMaxPeek(t *TestCase, min, max, peek int, stack *MinM
  if min != stack.GetMin() {
    t.Fail()
  }
  if max != stack.GetMax() {
    t.Fail()
  }
  if peek != stack.Peek() {
    t.Fail()
  }
}

func assertEqual(t *TestCase, a, b int) {
  if a != b {
    t.Fail()
  }
}

func (s *TestSuite) TestCase1(t *TestCase) {
  stack := NewMinMaxStack()
```

Run or submit code when you're ready.