Our Solution(s) Run Code

Solution 1

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

class Program {
    class MinHeap {
        var heap = [Int]()

        init(array: [Int]) {
            var array = array
            heap = buildHeap(array: &array)
        }

        // O(n) time | O(1) space
        func buildHeap(array: inout [Int]) -> [Int] {
            var firstParentIndex = Double((array.count - 2) / 2)
            firstParentIndex = firstParentIndex.rounded(.down)

            for var currentIndex in (0 ... Int(firstParentIndex)
                var endIndex = array.count - 1
                siftDown(currentIndex: &currentIndex, endIndex:
            }

            return array
        }

        // O(log(n)) time | O(1) space
        func siftDown(currentIndex: inout Int, endIndex: inout I
            var firstChildIndex = (2 * currentIndex) + 1
            while firstChildIndex < endIndex {
                var secondChildIndex = -1
                let potentialSecondChild = (2 * currentIndex) +
                if potentialSecondChild <= endIndex {
                    secondChildIndex = potentialSecondChild
                }
```

Your Solutions Run Code

Solution 1 Solution 2 Solution 3

```swift
class Program {
    class MinHeap {
        var heap = [Int]()

        init(array: [Int]) {
            var array = array
            heap = buildHeap(array: &array)
        }

        func buildHeap(array: inout [Int]) -> [Int] {
            // Write your code here.
            return []
        }

        func siftDown(currentIndex: inout Int, endIndex: inout
            // Write your code here.
        }

        func siftUp(currentIndex: inout Int, heap: inout [Int])
            // Write your code here.
        }

        func peek() -> Int {
            // Write your code here.
            return -1
        }

        func remove() -> Int {
            // Write your code here.
            return -1
        }

        func insert(value: Int) {
```

Our Tests Custom Output Submit Code

```
main :: IO ()
main = do
    let program = Program()
    runTests "Test Case 1" [] $ \Strings -> void $
        let heap = Program.MinHeap([3, 1, 6])
        in try testMinHeap(heap, heap)
    }
    runTests "Test Case 2" [] $ \Strings -> void $
        let heap = Program.MinHeap([1, 2, 3, 4, 5, 6,
        in try testMinHeap(heap, heap)
    }
    runTests "Test Case 3" [] $ \Strings -> void $
        let heap = Program.MinHeap([100, 11, 20, 7, 8,
        heap.insert(5)
        heap.remove()
        heap.remove()
        heap.insert(87)
        try testMinHeap(heap, heap)
    }
    runTests "Test Case 4" [] $ \Strings -> void $
        let heap = Program.MinHeap([-4, 5, 20, 8, -10
        try testMinHeap(heap, heap)
```

Run or submit code when you're ready.