Solution 1

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

using System.Collections.Generic;

public class Program {
  public class MinHeap {
    public List<int> heap = new List<int>();

    public MinHeap(List<int> array) {
      heap = buildHeap(array);
    }

    // O(n) time | O(1) space
    public List<int> buildHeap(List<int> array) {
      int firstParentIdx = (array.Count - 2) / 2;
      for (int currentIdx = firstParentIdx; currentIdx >= 0; currentIdx--) {
        siftDown(currentIdx, array.Count - 1, array);
      }
      return array;
    }

    // O(log(n)) time | O(1) space
    public void siftDown(int currentIdx, int endIdx, List<int> heap) {
      int childOneIdx = currentIdx * 2 + 1;
      while (childOneIdx <= endIdx) {
        int childTwoIdx = currentIdx * 2 + 2 <=
          endIdx ? currentIdx * 2 + 2 : -1;
        int idxToSwap;
        if (childTwoIdx != -1 && heap[childTwoIdx] < heap[childOneIdx]) {
          idxToSwap = childTwoIdx;
        } else {
          idxToSwap = childOneIdx;
        }
        if (heap[idxToSwap] < heap[currentIdx]) {
          swap(currentIdx, idxToSwap, heap);
          currentIdx = idxToSwap;
          childOneIdx = currentIdx * 2 + 1;
        } else {
          return;
        }
      }
    }

    // O(log(n)) time | O(1) space
    public void siftUp(int currentIdx, List<int> heap) {
      int parentIdx = (currentIdx - 1) / 2;
      while (currentIdx > 0 && heap[currentIdx] < heap[parentIdx]) {
        swap(currentIdx, parentIdx, heap);
        currentIdx = parentIdx;
        parentIdx = (currentIdx - 1) / 2;
      }
    }

    public int Peek() {
      return heap[0];
    }

    public int Remove() {
      swap(0, heap.Count - 1, heap);
      int valueToRemove = heap[heap.Count - 1];
      heap.RemoveAt(heap.Count - 1);
      siftDown(0, heap.Count - 1, heap);
      return valueToRemove;
    }

    public void Insert(int value) {
      heap.Add(value);
      siftUp(heap.Count - 1, heap);
    }

    public void swap(int i, int j, List<int> heap) {
      int temp = heap[j];
      heap[j] =  heap[i];
      heap[i] =  temp;
    }
  }
}

```

Solution 1 | Solution 2 | Solution 3

```
using System.Collections.Generic;

// Do not edit the class below except for the buildHeap,
// siftDown, siftUp, Peek, Remove, and Insert methods.
// Feel free to add new properties and methods to the class.
public class Program {
  public class MinHeap {
    public List<int> heap = new List<int>();

    public MinHeap(List<int> array) {
      heap = buildHeap(array);
    }

    public List<int> buildHeap(List<int> array) {
      // Write your code here.
      return null;
    }

    public void siftDown(int currentIdx, int endIdx, List<int> heap) {
      // Write your code here.
    }

    public void siftUp(int currentIdx, List<int> heap) {
      // Write your code here.
    }

    public int Peek() {
      // Write your code here.
      return -1;
    }

    public int Remove() {
      // Write your code here.
      return -1;
    }

    public void Insert(int value) {
      // Write your code here.
    }
  }
}

```

Run or submit code when you're ready.