Solution 1

```
class Heap {
  constructor(comparisonFunc, array) {
    this.heap = this.buildHeap(array);
    this.comparisonFunc = comparisonFunc;
    this.length = this.heap.length;
  }

  buildHeap(array) {
    const firstParentIdx = Math.floor((array.length - 2) / 2);
    for (let currentIdx = firstParentIdx; currentIdx >= 0; currentIdx--) {
      this.siftDown(currentIdx, array.length - 1, array);
    }
    return array;
  }

  siftDown(currentIdx, endIdx, heap) {
    let childOneIdx = currentIdx * 2 + 1;
    while (childOneIdx <= endIdx) {
      const childTwoIdx = currentIdx * 2 + 2 <= endIdx ? currentIdx * 2 + 2 : -1;
      let idxToSwap;
      if (childTwoIdx !== -1) {
        if (this.comparisonFunc(heap[childTwoIdx], heap[childOneIdx])) {
          idxToSwap = childTwoIdx;
        } else {
          idxToSwap = childOneIdx;
        }
      } else {
        idxToSwap = childOneIdx;
      }
      if (this.comparisonFunc(heap[idxToSwap], heap[currentIdx])) {
        this.swap(currentIdx, idxToSwap, heap);
        currentIdx = idxToSwap;
        childOneIdx = currentIdx * 2 + 1;
      } else {
        return;
      }
    }
  }

  siftUp(currentIdx, heap) {
    let parentIdx = Math.floor((currentIdx - 1) / 2);
    while (currentIdx > 0) {
      if (this.comparisonFunc(heap[currentIdx], heap[parentIdx])) {
        this.swap(currentIdx, parentIdx, heap);
        currentIdx = parentIdx;
        parentIdx = Math.floor((currentIdx - 1) / 2);
      } else {
        return;
      }
    }
  }

  peek() {
    return this.heap[0];
  }

  remove() {
    this.swap(0, this.length - 1, this.heap);
    const valueToRemove = this.heap.pop();
    this.length--;
    this.siftDown(0, this.length - 1, this.heap);
    return valueToRemove;
  }

  insert(value) {
    this.heap.push(value);
    this.length++;
    this.siftUp(this.length - 1, this.heap);
  }

  swap(i, j, heap) {
    const temp = heap[j];
    heap[j] = heap[i];
    heap[i] = temp;
  }
}

function MAX_HEAP_FUNC(a, b) {
  return a > b;
}

function MIN_HEAP_FUNC(a, b) {
  return a < b;
}

exports.ContinuousMedianHandler = ContinuousMedianHandler;
```