```
<include <iostream#
<include <vector#

} class MinHeap
:private
;std::vector<int> heap

} void heapifyUp(int index)
;int parent = (index - 1) / 2
} while (index > 0 && heap[index] < heap[parent])
;std::swap(heap[index], heap[parent])
;index = parent
;parent = (index - 1) / 2
{
{

} void heapifyDown(int index)
;int left = 2 * index + 1
;int right = 2 * index + 2
;int smallest = index

} if (left < heap.size() && heap[left] < heap[smallest])
;smallest = left
{
} if (right < heap.size() && heap[right] < heap[smallest])
;smallest = right
{

} if (smallest != index)
;std::swap(heap[index], heap[smallest])
;heapifyDown(smallest)
{
{

:public
{} ()MinHeap
```

```
    } void insert(int value)
        ;heap.push_back(value)
    ;heapifyUp(heap.size() - 1)
                                {

    } ()void removeMin
    } if (heap.empty())
            ;return
                                {
    ;()heap[0] = heap.back
        ;()heap.pop_back
        ;heapifyDown(0)
                                {

    } int getMin() const
    } if (!heap.empty())
        ;return heap[0]
                                {
return -1; // or throw an exception
                                {
                                ;{

                    } ()int main
                ;MinHeap minHeap
                ;minHeap.insert(5)
                ;minHeap.insert(3)
                ;minHeap.insert(8)
                ;minHeap.insert(1)

std::cout << "Minimum element: " << minHeap.getMin() << std::endl; // Output: 1

                ;()minHeap.removeMin
std::cout << "Minimum element after removal: " << minHeap.getMin() << std::endl; // Output: 3

                ;return 0
                                {
```