Heap Pseudocode

https://tutorialbit.com/data-structure/delete-an-element-from-heap/
https://www.tutorialspoint.com/minimum-element-in-a-max-heap-in-cplusplus

```
BEGIN deleteNode
    IF (heap is empty) THEN
        RETURN false
    ELSE
        size = sizeof(arr) / sizeof(arr[0])
        N = size(heap)
        minElement = heap[N/2]
        FOR i = n/2+1 until n DO
            minElement = min(minElement, heap[i]
        ENDFOR

        FOR i=1 until n
            IF (heap[i] = minElement) THEN
                M=i
                heap[i] = heap[M]
            ENDIF
        ENDFOR

        FOR i = m until n/2 DO
            IF (heap[2*i] > heap[(2*i)+1] && heap[2*i] > heap[i] THEN
                swap(heap[i], heap[2*i])
                i=2*i
            ELSE IF (heap[2*i] < heap[(2*i)+1]) && heap[(2*i)+1] > heap[i] THEN
                swap(heap[i], heap[(2*i)+1])
                i=(2*i)+1
            ELSE
                break
            ENDIF
        ENDFOR
        n=n-1
    ENDIF
END DELETENODE


BEGIN correctHeap
    index = 1
    WHILE (index < size) DO
        siftdown(heap, index)
        index++
```

```
    ENDWHILE
END correctHeap

BEGIN siftdown
    leftchildindex = root*2+1
    rightchildindex = root*2+2
    IF (leftchildindex <= last) THEN
        leftkey = heap[leftchildindex].key
        IF (rightchildindex <= last) THEN
            rightkey = heap[rightchildindex].key
        ELSE
            rightkey = leftkey-1
        ENDIF
        IF (leftkey > rightkey) THEN
            largerchildkey = leftkey
            largerchildindex = leftchildindex
        ELSE
            largerchildkey = rightkey
            largerchildindex = rightchildindex
        ENDIF
        IF (heap[root].key < largerchildkey) THEN
            swap(heap, root, largerchildindex)
            siftdown(heap, largerchildindex, last)
        ENDIF
    ENDIF
END siftdown
```