6.5-3.

Write pseudocode for the procedures Heap-Minimum, Heap-Extract-Min, Heap-Decrease-Key, and Min-Heap-Insert that implement a min-priority queue with a min-heap.

Answer.

The procedure Heap-Minimum implements the Minimum operation in $\Theta(1)$ time.

```
HEAP-MINIMUM(A)
    return A[1]
```

The procedure Heap-Extract-Min is implemented almost the same to Heap-Extract-Max except that it maintains the min-heap property after eliminating the node of smallest key.

```
HEAP-EXTRACT-MIN(A)
    if A.heap-size < 1
        error "heap underflow"
    min = A[1]
    A[1] = A[A.heap-size]
    A.heap-size = A.heap-size − 1
    MIN-HEAPIFY(A, 1)
    return min
```

Like Heap-Extract-Max, the running time of Heap-Extract-Min is also $O(\lg n)$.

The procedure Heap-Decrease-Key implements the Decrease-Key operation. It first decrease the key of element $A[i]$ to its new value, and as the min-heap property might be violated, it then traverse a simple path from this node toward the root to find a proper place for the newly decreased key.

```
HEAP-DECREASE-KEY(A, i, key)
    if key > A[i]
        error "new key is greater than current key"
    A[i] = key
    while i > 1 and A[PARENT(i)] > A[i]
        exchange A[i] with A[PARENT(i)]
        i = PARENT(i)
```

The running time of Heap-Decrease-Key on an n-element heap is $O(\lg n)$, which has been explained when we presented Heap-Increase-Key.

The procedure Min-Heap-Insert also implements the Insert operation. Like Max-Heap-Insert, the procedure first expands the max-heap by adding to the tree a new leaf whose key is $+\infty$. Then it calls Heap-Decrease-Key to set the key of this new node to its correct value and maintain the min-heap property.

```
MIN-HEAP-INSERT(A, key)
    A.heap-size = A.heap-size + 1
    A[A.heap-size] = +∞
    HEAP-DECREASE-KEY(A, A.heap-size, key)
```

The running time of Min-Heap-Insert on an n-element heap is $O(\lg n)$.

Email address: informlarry@gmail.com