6.2-5.

The code for MAX-HEAPIFY is quite efficient in terms of constant factors, except possibly for the recursive call in line 10, which might cause some compilers to produce inefficient code. Write an efficient MAX-HEAPIFY that uses an iterative control construct (a loop) instead of recursion.

Answer.

As Figure 6.2 shows, MAX-HEAPIFY successively lets a node “float down” in the max-heap as long as the subtree rooted at it violates the max-heap property.

```
MAX-HEAPIFY($A, i$)
$l$ = LEFT($i$)
$r$ = RIGHT($i$)
$largest$ = index of MAX($A[i], A[l], A[r]$)
while $r \leq A.heap\text{-}size$ and $largest \neq i$
    exchange $A[i]$ with $A[largest]$
    $i = largest$
    $l$ = LEFT($i$)
    $r$ = RIGHT($i$)
    $largest$ = index of MAX($A[i], A[l], A[r]$)
```