Exercises

6.1-1
What are the minimum and maximum numbers of elements in a heap of height h?

6.1-2
Show that an n-element heap has height $\lfloor \lg n \rfloor$.

6.1-3
Show that in any subtree of a max-heap, the root of the subtree contains the largest value occurring anywhere in that subtree.

Chapter 6   Heapsort

6.1-4
Where in a max-heap might the smallest element reside, assuming that all elements are distinct?

6.1-5
Is an array that is in sorted order a min-heap?

6.1-6
Is the array with values $\langle 23, 17, 14, 6, 13, 10, 1, 5, 7, 12 \rangle$ a max-heap?

6.1-7
Show that, with the array representation for storing an n-element heap, the leaves are the nodes indexed by $\lfloor n/2 \rfloor + 1, \lfloor n/2 \rfloor + 2, \ldots, n$.

6.2-1
Using Figure 6.2 as a model, illustrate the operation of MAX-HEAPIFY$(A, 3)$ on the array $A = \langle 27, 17, 3, 16, 13, 10, 1, 5, 7, 12, 4, 8, 9, 0 \rangle$.

6.2-2
Starting with the procedure MAX-HEAPIFY, write pseudocode for the procedure MIN-HEAPIFY(A, i), which performs the corresponding manipulation on a min-heap. How does the running time of MIN-HEAPIFY compare to that of MAX-HEAPIFY?

6.2-3
What is the effect of calling MAX-HEAPIFY(A, i) when the element $A[i]$ is larger than its children?

6.2-4
What is the effect of calling MAX-HEAPIFY(A, i) for $i > A.heap\text{-}size/2$?

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.

6.2-6
Show that the worst-case running time of MAX-HEAPIFY on a heap of size n is $\Omega(\lg n)$. (*Hint:* For a heap with n nodes, give node values that cause MAX-HEAPIFY to be called recursively at every node on a simple path from the root down to a leaf.)