As with previous chapters, we are going to lay out the learning objectives for this chapter, and split them into what you should get on the first reading vs the second reading.

1 First Reading: Understand Key Ideas

On your first reading, the goal is to get the big ideas, and build the mental scaffolding in which to hold the details on your second reading—the learning objectives for your first reading are generally **Remember** and **Understand** objectives. If you are missing a few details (but not completely lost), it is ok to move ahead, and come back to them the next time through—you might ask questions on the course forums too! Note you should **sleep at least one night between the first and second reading** to give your brain time to process the information.

- **Heaps and Priority Queues: Intro**
  - **Define** *priority queue*.
- **Heaps Concepts**
  - **Recall** *complete binary tree* (from Chapter 22).
  - **Explain** the heap ordering rule.
  - **Define** *heap*, *max-heap*, and *min-heap*.
  - **Explain** how to implement the `peek` operation for a priority queue with a heap.
  - **Explain** how to add an item to a heap.
  - **Explain** why the `dequeue` operation of a priority queue implemented with a heap would remove the root element.
  - **Explain** how to remove the root element from a heap.
- **Heaps Array Implementation**
  - **Define** *sentinel*.
  - **Explain** why a sentinel can be useful in a heap.
  - **Explain** why in some cases it is easy to pick a sentinel value, but in others it is impossible.
  - **Explain** how the conceptually tree-like structure of a heap is mapped onto an array.
  - **Explain** how we can reduce the number of cases required in bubbling down.
- **STL Priority Queue**
  - **Name** the STL class that implements a priority queue.
  - **Explain** how to use STL's priority queue if you want the smallest element to have the highest priority.
  - **Explain** how to use STL's priority queue if you want to write your own custom ordering.
- **Priority Queue Use: Compression**
  - **Define** *compression algorithm*.
  - **Explain** how Huffman coding works.

2 Second Reading

As always, we strongly recommend you sleep between your first and second reading. The second reading is where you want to focus on the higher-level learning objectives, which build on the lower-level learning objectives you worked on in the first reading. Sleeping gives your brain a chance to process the information from the first reading.

- **Show** how to add to or remove from a heap in its conceptual tree representation.
- **Show** how the tree representation of a heap maps onto an array.
- **Show** how to add to or remove from a heap in its array representation.
- **Implement** a heap or priority queue.
- **Assess** if a priority queue is an efficient ADT for the problem you are solving.
- **Write** code that uses a priority queue.
- **Show** how a Huffman tree is built from input symbol frequencies.
- **Show** how a Huffman tree produces the encoding for a symbol.
- **Write** code to perform Huffman compression. (Note: you will do this broken down into 4 assignments on the MLP).