Here is the ADT specification of an unordered list from Section 7.2.

- `UnorderedList()` returns an empty list.
- `isEmpty()` returns `True` if the list is empty and `False` otherwise.
- `length()` returns the number of items in the list.
- `add(item)` adds the given item to the list. An item may occur more than once in the list.
- `search(item)` returns `True` if the given item is in the list and `False` otherwise.
- `remove(item)` assumes that the given item is in the list, and removes it from the list. If the item occurs more than once in the list, then one of its occurrences is removed.

In Assignment 3 we add new methods to the unordered list. In doing so we change the specification significantly. The enhanced list supports a notion of indexing, so that there is a definite first item, second item, and so on. Notice that no such notion of indexing appears in the unordered list specification above. The enhanced list is really more similar to built-in Python lists, than to the unordered list above.

Here is the ADT specification of a stack from Section 2.3. As is always the case in ADT specifications, there is no mention of how we might implement the stack; there is only a description of the stack's capabilities.

- `Stack()` returns an empty stack.
- `isEmpty()` returns `True` if the stack is empty and `False` otherwise.
- `size()` returns the number of items in the stack.
- `push(item)` adds the given item to the top of the stack.
- `pop()` removes the top item from the stack and returns it.
- `peek()` returns the top item on the stack without modifying the stack. (Note: Many authors omit this method from the stack specification.)

Here is the ADT specification of a queue from Section 2.4.

- `Queue()` returns an empty queue.
- `isEmpty()` returns `True` if the queue is empty and `False` otherwise.
- `size()` returns the number of items in the queue.
- `enqueue(item)` adds the given item to the end of the queue.
- `dequeue()` removes the first item from the queue and returns it.