Lab 4: Abstract Data Type Solution

2) Queues

1. Implement *Queue* class found in *myqueue.py*

```
...
class Queue:
    """A first-in-first-out (FIFO) queue of items.

    Stores data in a first-in, first-out order. When removing an item from the
    queue, the least recently-added item (i.e. the oldest item in the Queue)
    is the one that is removed.
    # === Private Attributes ===
    # _items:
    #     The items stored in this queue. The front of the list represents
    #     the front of the queue.
    """
    _items: List
    def __init__(self) -> None:
        """Initialize a new empty queue."""
        self._items = []

    def is_empty(self) -> bool:
        """Return whether this queue contains no items.

        >>> q = Queue()
        >>> q.is_empty()
        True
        >>> q.enqueue('hello')
        >>> q.is_empty()
        False
        """
        return self._items == []

    def enqueue(self, item: Any) -> None:
        """Add <item> to the back of this queue.
        """
        self._items.append(item)

```

```
35          def dequeue(self) -> Optional[Any]:
36              """Remove and return the item at the front of this queue.
37
38              Return None if this Queue is empty.
39              (We illustrate a different mechanism for handling an
    erroneous case.)
40
41              >>> q = Queue()
42              >>> q.enqueue('hello')
43              >>> q.enqueue('goodbye')
44              >>> q.dequeue()
45              'hello'
46              """
47              if self.is_empty():
48                  raise EmptyStackError
49              else:
50                  return self._items.pop(0)
51      ...
52
```

Listing 1: task_2_q1_solution.py

2. Complete functions *product* and *product_star* in *myqueue.py*