The Queue ADT

- Ordered collection/container**, first-in-first-out** (FIFO) (i.e., removal in same order as insertion)
- Functionality:
    - initialize
    - insert item at rear of queue (i.e., enqueue or push)
    - remove item from front of queue (i.e., dequeue or pop)
    - management: size, isEmpty, etc.

Class Invariant

- Explicit statement of how the data structure is used to represent the ADT

- Queue class implemented with (pointer-based) <u>linked list</u> for *data*

  (1) No. of items in queue stored in member variable *count*
  (2) Items are stored in a linked list, with the front of the queue stored at head node, and the rear of the queue stored at the tail node.
  (3) Head pointer of the linked list is *frontPtr*. For a non-empty queue, *rearPtr* is tail pointer of the linked list; for empty queue, *rearPtr* and *frontPtr* NULL

  Enqueue/Push: ***insertNodeAtRear*** which is O(1) since we maintain *rearPtr*

  Dequeue/Pop: ***removeNodeAtHead*** which is O(1) since we maintain *frontPtr*

- Queue class implemented with <u>array</u> for *data*

  (1) No. of items in queue stored in member variable *count*
  (2) For a non-empty queue, items are stored in a 'circular' array beginning at *data[first]* and continuing through *data[last].* Total capacity of array is *CAPACITY*
  (3) For an empty queue, last is some valid index and first is *nextIndex(last)* (i.e., the index position that comes after *last*; may not be *last* + 1)

  Constructor: set *count* and *first* to 0, set *last* to CAPACITY-1 O(1)

  Next Index: return ((index + 1) % CAPACITY) O(1)

  Enqueue/Push: set *last* to nextIndex(*last*), set *data[last]* to entry, and increment *count* O(1)

  Dequeue/Pop: set *first* to nextIndex(first) and decrement *count* O(1)

Overflow and Underflow Errors

- *Queue overflow*: attempting to add an entry to a full queue
- *Queue underflow*: attempting to remove an entry from an empty queue

Standard Template Library (STL) Queue Class

- *#include <queue>*
- queue<int> myQueue;

The Priority Queue ADT

- Ordered collection/container, **highest priority entry removed first**
- Linked list and array implementations possible, but operations can be O(n)

Standard Template Library (STL) Priority Queue Class

- *#include <queue>*
- priority_queue<int> myPQueue;