Queues: ADT and implementations

- queues store a set of data values, maintaining the order in which they were entered
- new values are inserted at the back
- values are removed from the front
- generally used when we want to process items in the order they arrived
- referred to as FIFO, first in first out
- implementations often very similar to linked list
- circular buffers are another common implementation

Queue ADT

- each element being stored might consist of multiple fields (like with lists)
- operations to insert, typically called enqueue
- operations to remove, typically called dequeue
- possible additional operations to print the queue, check the size, find/remove something that's in the queue

Common uses

- generally used when we're processing things in the order they arrive, e.g.
  - handling sales: might use a queue of purchase orders/requests
  - handling jobs sent to a printer: might use queue of the print requests
- common variation is the priority queue, in which jobs of higher priority are done before jobs of lower priority
  - jobs of equal priority are handled in traditional queue order

Implementation: list style

- can use our linked list code almost verbatim
- maintain pointers to front and back
- restrictions
  - we always insert at the back
  - we always remove from the front

Implementation: array style

- can implement a queue using an array
- if we're removing from front, then either
  - we have to shuffle the array elements over one position after each remove (to keep the front in array position 0)
  - or the position of front changes over time, e.g.
    - original front element in array position 0
    - after one remove the front is the element in position 1
    - after two removes it's the one in position 2, etc
- latter approach leads to the “circular buffer” concept
  - csci.viu.ca/~wesselsd/courses/csci161/slides/circularBuffer.pdf