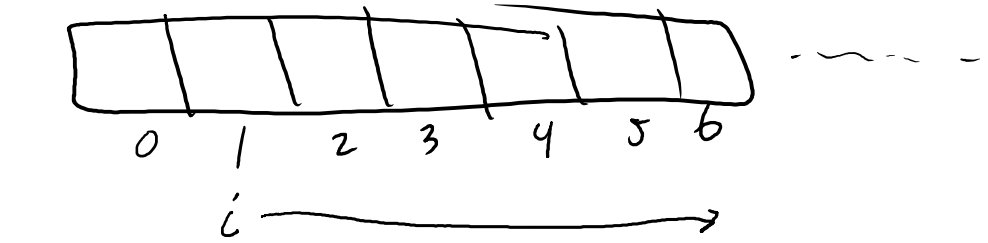
Queues

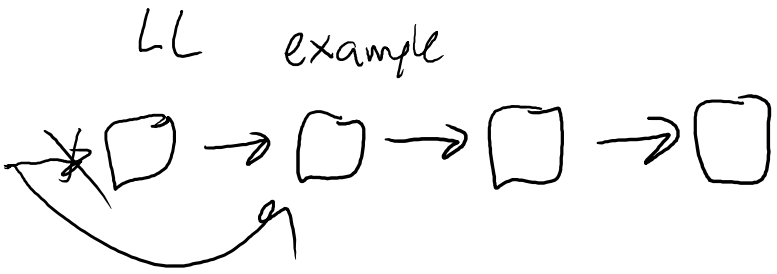
Tuesday, January 30, 2018 3:41 PM

- Like stacks, queues put strict requirements on adding and removing data
- Rule:
  - First in - first out (FIFO)
  - Last in - last out (LILO)

- Real world example:
  - Line (waiting to buy food, ride an amusement park ride, traffic, etc.)
- Computer examples:
  - Internet traffic
  - File processing
  - Printing
- Queue operations:
  - Enqueue - add to the end of a queue. Sometimes also called push.
  - Dequeue - removing from the front of a queue. Sometimes also called pop.

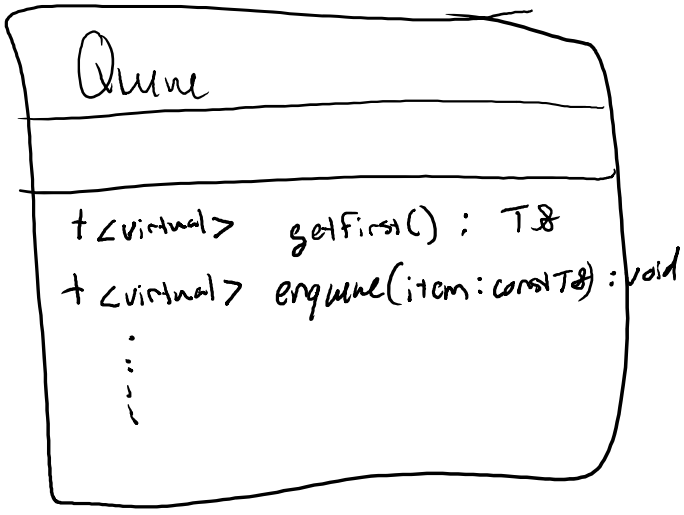
Code Question: Where should our loop range from (vector queue)?





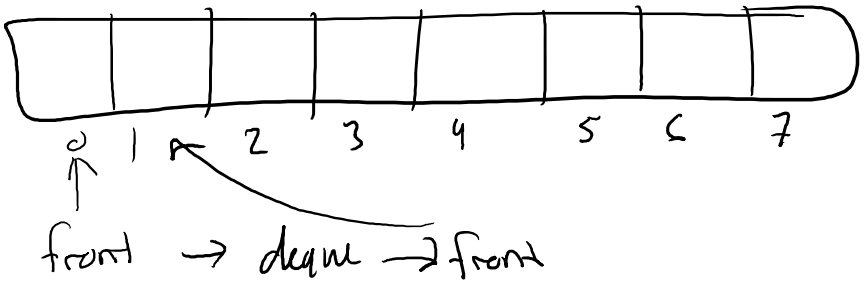
Queue ADT as UML

```
class Queue
{
public:
        virtual T &getFirst() = 0;
   virtual const T &getFirst() const = 0;
        virtual void enqueue(const T &item) = 0;
        virtual T dequeue() = 0;
   virtual int getSize() const = 0;
};
```



What's the issue with vector-based queues?

- Dequeues are slow relative to LL - O(N) vs O(1)
- It's slow because we maintain a fixed "front" of the queue (element 0)
- What happens if we maintain a "floating" front



- What is the efficiency of maintaining a logical (floating) front? O(1)
- What's the downside of maintaining a logical front?
  - Lots of wasted memory. If we do 1M enqueues and 1M dequeues, our queue is empty even though vector is of size 1M
- What happens if we maintain a logical (floating) end?