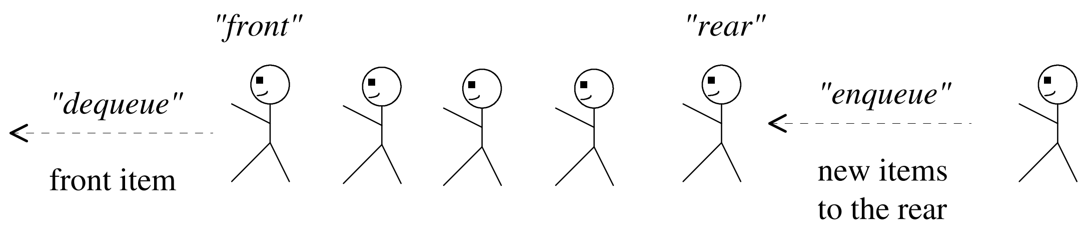
Data Structures Name:____________________________

A FIFO *queue* is basically what we think of as a waiting line.



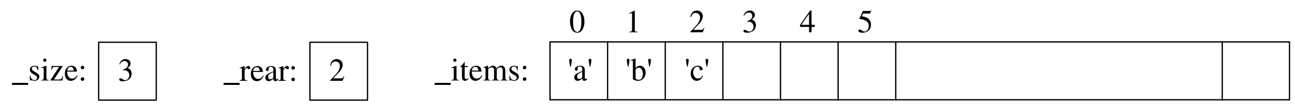
The operations/methods on a queue object, say myQueue are:

Method Call on myQueue object	**Description**
`myQueue.dequeue()`	Removes and returns the front item in the queue.
`myQueue.enqueue(myItem)`	Adds `myItem` at the rear of the queue
`myQueue.peek()`	Returns the front item in the queue without removing it.
`myQueue.isEmpty()`	Returns `True` if the queue is empty, or `False` otherwise.
`len(myQueue)`	Returns the number of items currently in the queue
`str(myQueue)`	Returns the string representation of the queue

1. Complete the following table by indicating which of the queue operations should have preconditions. Write "none" if a precondition is not needed.

Method Call on myQueue object	**Precondition(s)**
`myQueue.dequeue()`	
`myQueue.enqueue(myItem)`	
`myQueue.peek()`	
`myQueue.isEmpty()`	
`len(myQueue)`	
`str(myQueue)`	

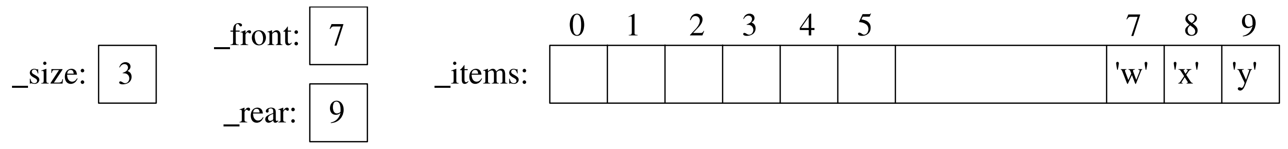
One possible implementation of a queue would be to use an Array _items to store the queue items such that
- the front item is always stored at index 0,
- an integer _size data-attribute is used to maintain the number of items in the queue
- an integer _rear data-attribute is used to maintain the index of the rear item



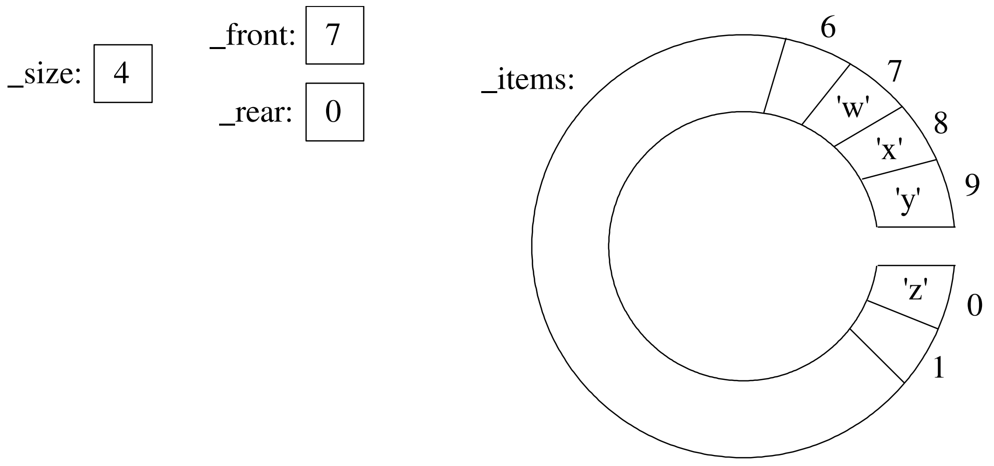
a) What would be the big-oh notation for enqueue?

b) What would be the big-oh notation for dequeue?

As pointed out in section 15.4.2 we can avoid "shifting the items left" on a dequeue operation by maintaining the index of the front item in addition to the rear. Overtime, the used portion of the array (where the actual queue items are) will drift to the right end of the array with the left end being unused, i.e.:



Now if we enqueue another item, we'd like the rear of the queue to "wrap" around to index 0, i.e., we'd like the array to behave "circularly." After we enqueue('z'), we would have:

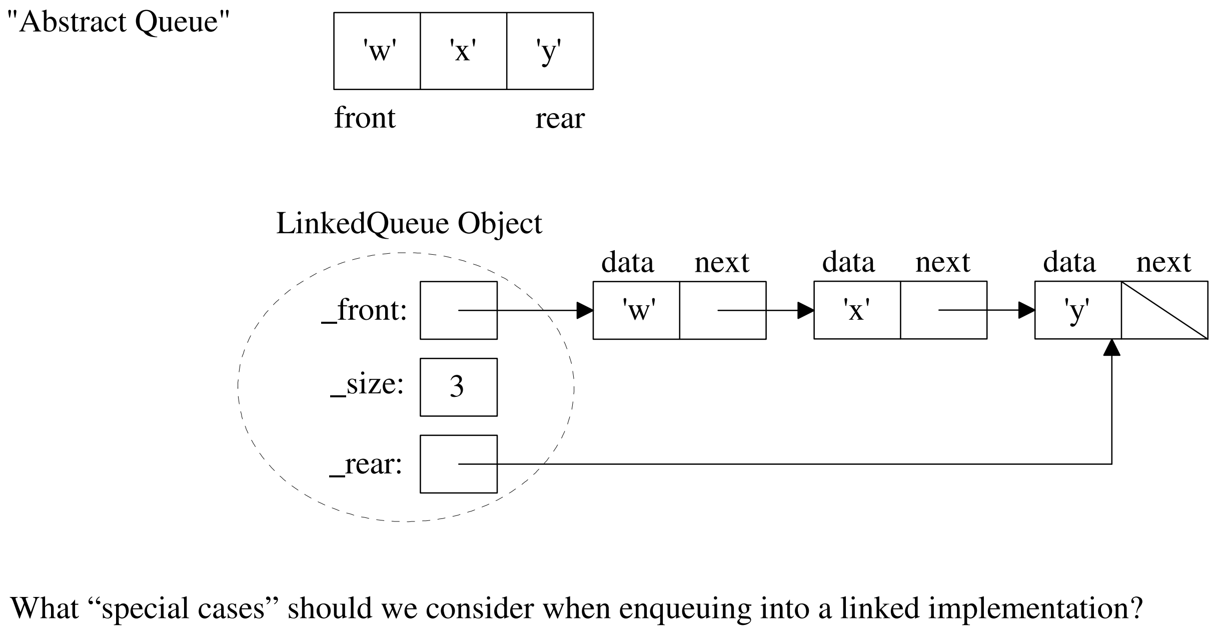


c) How would we count "circularly" to wrap back around to 0?

d) What would be the big-oh notation for enqueue using a "circular array" implementation?

e) What would be the big-oh notation for dequeue using a "circular array" implementation?

2. A singly-linked list implementation of the queue (`LinkedQueue` class in the text). Conceptually, a `LinkedQueue` object would look like:



a) What "special cases" should we consider when enqueuing into a linked implementation?

b) What would be the steps for the “normal” case?

c) Would the code for the “normal” case work for any of the special cases?