1. 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

   
   _items: 
   0 1 2 3 4 5
   'a' 'b' 'c' | |
   _size: 3  _rear: 2

   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.:

   _items: 
   0 1 2 3 4 5
   'w' 'x' 'y' | |
   _size: 3  _front: 7  _rear: 9

   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:

   _items: 
   0 1 2 3 4 5 6 7 8 9
   'w' 'x' 'y' 'z' | |
   _size: 4  _front: 7  _rear: 0

   c) How would we count “circularly”?

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

   e) What would be the big-oh notation for dequeue?
2. A singly-linked list implementation of the queue (LinkedQueue class in the text). Conceptually, a LinkedQueue object would look like:

"Abstract Queue"

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>'w'</td>
<td>'x'</td>
<td>'y'</td>
</tr>
</tbody>
</table>

front  rear

LinkedQueue Object

_a.front:  _size:  _rear:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>'w'</td>
<td></td>
</tr>
<tr>
<td></td>
<td>'x'</td>
<td></td>
</tr>
<tr>
<td></td>
<td>'y'</td>
<td></td>
</tr>
</tbody>
</table>

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?
3. A priority queue has the same operations as a regular queue, except the items are NOT returned in the FIFO (first-in, first-out) order. Instead, each item has a priority that determines the order they are removed. A hospital emergency room operates like a priority queue -- the person with the most serious injury has highest priority even if they just arrived.

a) Suppose that we have a priority queue with integer priorities such that the smallest integer corresponds to the highest priority. For the following priority queue, which item would be dequeued next?

```
priority queue:
  40  10  79
  30  13

```

b) To implement a priority queue, we could use an unordered Python list. If we did, what would be the worst-case theta \( \Theta( ) \) notation for each of the following methods: (justify your answer)

- **enqueue:**
- **dequeue:**

c) To implement a priority queue, we could use a linked list ordered by priorities, e.g., the LinkedPriorityQueue class of chapter 15. If we did, what would be the worst-case theta \( \Theta( ) \) notation for each of the following methods: (justify your answer)

- **enqueue:**
- **dequeue**

4. Sections 18.9 - 18.11 discuss a very “non-intuitive”, but powerful list/array-based approach to implement a priority queue, call a heap. The list/array is used to store a complete binary tree (a full tree with any additional leaves as far left as possible) with the items being arranged by heap-order property, i.e., each node is less than either of its children. An example of a heap “viewed” as a complete binary tree would be:

```
6 [0]
  /  \
 [1]  [2]
  /  \
 [3]  [4]
  /  \
15 114 [5]
    /  \  |
    |     |
  300 125 50
```
a) For the above heap, the list/array indexes are indicated in [ ]'s. For a node at index $i$, what is the index of:
- its left child if it exists:
- its right child if it exists:
- its parent if it exists:

b) What would the above heap look like after adding 13 and 8?

c) What is the worst-case theta ($\Theta(\cdot)$) notation for adding an new item in the heap?