1. A “normal” array implementation of the stack:

Using an array implementation would look something like:

```
stackArray: a b c
0     1      2    3 99
```

top: 2 stackSize: 100

Complete the big-oh notation for each of the following stack methods assuming an array implementation:

<table>
<thead>
<tr>
<th>Method</th>
<th>Big-oh</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constructor</td>
<td></td>
</tr>
<tr>
<td>isFull()</td>
<td></td>
</tr>
<tr>
<td>isEmpty()</td>
<td></td>
</tr>
<tr>
<td>size()</td>
<td></td>
</tr>
<tr>
<td>peek()</td>
<td></td>
</tr>
<tr>
<td>pop()</td>
<td></td>
</tr>
<tr>
<td>push(item)</td>
<td></td>
</tr>
</tbody>
</table>

2. Consider a "normal" linked implementation of the stack:

```
myStack
value    next value    next value    next
a     b     c
```

top: c itemCount: 3

Complete the big-oh notation for each of the following stack methods assuming this linked implementation:

<table>
<thead>
<tr>
<th>Method</th>
<th>Big-oh</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constructor</td>
<td></td>
</tr>
<tr>
<td>isFull()</td>
<td></td>
</tr>
<tr>
<td>isEmpty()</td>
<td></td>
</tr>
<tr>
<td>size()</td>
<td></td>
</tr>
<tr>
<td>peek()</td>
<td></td>
</tr>
<tr>
<td>pop()</td>
<td></td>
</tr>
<tr>
<td>push(item)</td>
<td></td>
</tr>
</tbody>
</table>

3. One possible implementation of a queue would be to use an queueArray to store the queue items such that

- the front item is always stored at index 0,
- an integer numItems maintains the number of items in the queue
- an integer rear maintains the index of the rear item
- an integer queueSize maintains the size of the array

```
queueArray: 'a' 'b' 'c'
0     1     2     3     4     5     99
```

numItems: 3 rear: 2

queueSize: 100

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

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

```
<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>96</th>
<th>97</th>
<th>98</th>
<th>99</th>
</tr>
</thead>
</table>
queueArray: |   |   |   |   |   | 'm' | 's' | 'b' | 'j' |
```

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:

```
<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>96</th>
<th>97</th>
<th>98</th>
<th>99</th>
</tr>
</thead>
</table>
queueArray: |   |   | 's' | 'b' | 'j' | 'z' |   |    |    |
```

2. A singly-linked list implementation of the queue would conceptually look like:

```
<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
</table>
LinkedQueue Object: |   |   |   |   |   |   |   |   |   |   |
```

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 emergence 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 array. If we did, what would be the worst-case theta (Θ( )) 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. If we did, what would be the worst-case theta (Θ( )) notation for each of the following methods: (justify your answer)

- enqueue:
- dequeue
4. A very “non-intuitive”, but powerful array-based approach to implement an priority queue, call a heap. An 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 or equal to either of its children. An example of a heap “viewed” as a complete binary tree would be:

```
[0]
[1] 6
[2] 15
[3] 114
[4] 10
[5] 20
[6] 20
[7] 50
[8] 300
[9] 125
[10] 117
```

(a) For the above heap, the 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) Items are added as a “new” leaf in the complete binary tree. To restore the heap property, they are repeatedly compared with their parent and “sifted up” the heap to their correct spot. What would the above heap look like after adding 13 and 8?