A Deque (pronounced "Deck") is a linear data structure which behaves like a double-ended queue, i.e., it allows adding or removing items from either the front or the rear of the Deque.

1. One possible implementation of a Deque would be to use a Python list to store the Deque items such that
   - the rear item is **always stored at index 0**,
   - the front item is always stored at the highest index (or -1)

   ```python
   class Deque(object):
       def __init__(self):
           self.items = list()
   ``

   a) Complete the `__init__` method and determine the big-oh, \( O() \), for each Deque operation, assuming the above implementation. Let \( n \) be the number of items in the Deque.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>isEmpty</td>
<td>( O(1) )</td>
</tr>
<tr>
<td>addFront</td>
<td>( O(1) )</td>
</tr>
<tr>
<td>removeFront</td>
<td>( O(1) )</td>
</tr>
<tr>
<td>addRear</td>
<td>( O(n) )</td>
</tr>
<tr>
<td>removeRear</td>
<td>( O(n) )</td>
</tr>
<tr>
<td>size</td>
<td>( O(1) )</td>
</tr>
</tbody>
</table>

   b) Write the methods for the `addRear` and `removeRear` operations.

   ```python
def addRear(self, newItem):
    self.items.insert(0, newItem)
def removeRear(self):
    return self.items.pop(0)
```

2. An alternative implementation of a Deque would be a linked implementation as in:

   ```python
   class LinkedDeque(object):
       def __init__(self):
           self._rear = None
           self._front = None
           self._size = 0
   ``

   a) Complete the `__init__` method and determine the big-oh, \( O() \), for each Deque operation assuming the above linked implementation. Let \( n \) be the number of items in the Deque.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>isEmpty</td>
<td>( O(1) )</td>
</tr>
<tr>
<td>addFront</td>
<td>( O(1) )</td>
</tr>
<tr>
<td>removeFront</td>
<td>( O(n) )</td>
</tr>
<tr>
<td>addRear</td>
<td>( O(1) )</td>
</tr>
<tr>
<td>removeRear</td>
<td>( O(1) )</td>
</tr>
<tr>
<td>size</td>
<td>( O(1) )</td>
</tr>
</tbody>
</table>

   b) Suggest an improvement to the above linked implementation of the Deque to speed up some of its operations.

   Use a doubly-linked list of nodes
Data Structures (CS 1520)  
Lecture 6  
Name: 

from node import Node

class Node2Way(Node):
    def __init__(self, initdata):
        Node.__init__(self, initdata)
        self.previous = None

def getPrevious(self):
    return self.previous

def setPrevious(self, newprevious):
    self.previous = newprevious

3. An alternative implementation of a Deque would be a doubly-linked implementation as in:

DoublyLinkedDeque Object

- isEmpty: False
- addFront: ('a')
- removeFront: (None)
- addRear: ('b')
- removeRear: (None)
- size: 3

a) Determine the big-oh, \( O() \), for each Deque operation assuming the above doubly-linked implementation. Let \( n \) be the number of items in the Deque.

<table>
<thead>
<tr>
<th></th>
<th>isEmpty</th>
<th>addFront</th>
<th>removeFront</th>
<th>addRear</th>
<th>removeRear</th>
<th>size</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( O(1) )</td>
<td>( O(1) )</td>
<td>( O(1) )</td>
<td>( O(1) )</td>
<td>( O(1) )</td>
<td>( O(1) )</td>
</tr>
</tbody>
</table>

4. 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:

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

- enqueue: \( O(1) \)
- dequeue: \( O(n) \)

c) To implement a priority queue, we could use a Python list order by priorities in descending order. If we did, what would be the big-oh notation for each of the following methods: (justify your answer)

- enqueue: \( O(n) \)
- dequeue: \( O(1) \)
DoublyLinkedDeque removeFront (normal case)

if self._size == 0:
    raise ValueError("Cannot remove front from empty deque")

1. temp = self._front

2. self._front = temp.getPrevious()
   if self._size == 1:
       self._rear = None
   else:
       self._front.setNext( None )

3. self._size -= 1

4. self._size = self._size - 1

5. return temp.getData()

Special cases:

1. empty Deque: precondition
   raise Exception

2. remove only item in Deque

   \[ \text{Diagram} \]