CPE 202

Lab 3: Queue Implementations

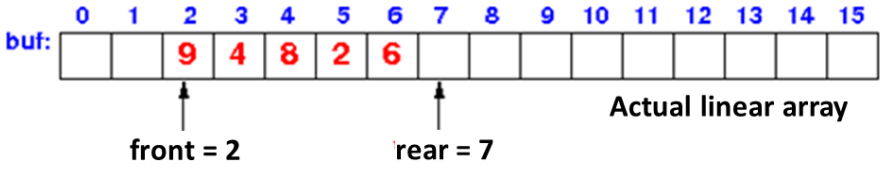
Goals:

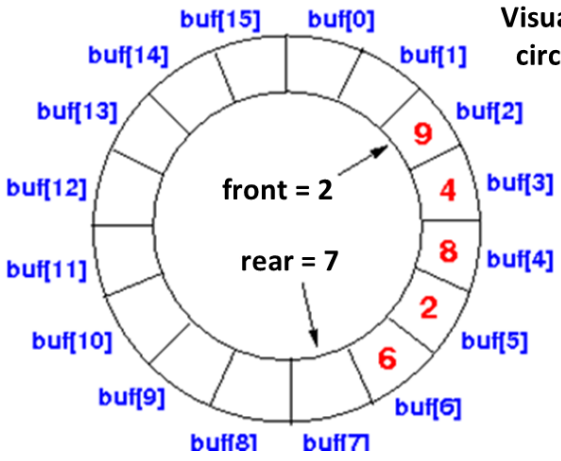
- **Implement a Queue class using the NodeList structure: queue_nodelist.py**
- **Implement a Queue class using a circular array: queue_array.py**

Before doing this lab make sure to read over section 4.10, 4.11, and 4.12 carefully. You will not be doing the implementation described in section 4.12, but it will help you understand the underlying concepts.

- For the NodeList implementation, you'll be using the Node class defined in the queue_nodelist.py module. In order to achieve O(1) behavior for the enqueue and dequeue operations, the Queue wrapper class will keep track of two NodeLists. We'll refer to these NodeLists as the "rear" NodeList and the "front" NodeList. When items are enqueued, they are simply added to the "rear" NodeList. When an item is dequeued, the first item from the "front" NodeList is removed. If there are no items in the "front" NodeList, we remove items from the "rear" NodeList and add them to the "front" NodeList, until the "rear" NodeList is empty, after which can remove the first item in the "front" NodeList to complete the dequeue operation. The process of moving the items from the "rear" NodeList to the "front" NodeList will reverse the order of the items, resulting in the FIFO behavior of a Queue.

- The second implementation will use a circular array for storing the items in the queue. There are different ways of doing this, but they share the idea presented in the picture below. Items are added to the rear of the queue using the next “free entry” in an array. Items are removed from the front of the queue. The indices front and rear are incremented as items are added and deleted from the queue, and the indices “wrap around” when they reach the end of the array. Why is using a circular array better than what the text does (i.e. inserting new items at index 0 of array, and removing items from the end of the array)?





For both Queue implementations:

- Attempting to dequeue an item from an empty Queue will raise an IndexError
- All methods must have ***O(1)*** performance (the speed must not be affected by the size of the queue)

For the array implementation:

- Attempting to enqueue an item into a full Queue will raise an IndexError
- As with your stack array implementation, you **may NOT use** any of the following Python List operations:
  - `append()`
  - `insert()`
  - `extend()`
  - `remove()`
  - `pop()`
  - `+` (concatenations)
  - List slicing (e.g. `some_list[2:9]`)

The following starter files will be in your GitHub repository. Complete the implementations and ensure that your implementations are committed and pushed back to your repository.

- **`queue_array.py`**: Contains an array (Python List) based implementation of the **Queue** class
- **`queue_nodelist.py`**: Contains a linked based implementation of the **Queue** class
- **`queue_array_tests.py`:** Contains your set of thorough tests to ensure your implementations work correctly for the array implementation.
- **`queue_nodelist_tests.py`:** Contains your set of thorough tests to ensure your implementations work correctly for the linked list implementation.