Chapter 04, 4.5 – Link-Based Queue Implementations, Review Problem(s): ***19, 20, 23, 26***

Chapter 04, 4.6 – Queue Applications: Palindromes, Review Problem(s): ***27***

Chapter 04, 4.6 – Queue Variations, Review Problem(s): ***31, 33***

Each implementation of the Queue ADT will have the same basic methods. But it is interesting to compare how a Link based implementation differs from an Array based implementation.

It is interesting to consider the differences between a standard linked queue, which has both ***front*** and ***rear*** instance variables, and a circular linked queue, which just needs a ***rear*** instance variable.

Chapter 04 PPT, pg 25-49.

How would you use both a queue and a stack to determine if a given input String is a palindrome?

- Take each letter and push it on a stack and enqueue it in a queue.
- While the stack and queue are not empty and palindrome=true
    - Pop the stack and dequeue the queue
    - Compare these letters
        - If they are not equal, set palindrome=false
- Return the palindrome boolean

How would you modify the LinkedQueue.java (in Moodle) to implement a CircularLinkedQueue? ***Easy ones first.***

```
  public boolean isEmpty()
  // Returns true if this queue is empty; otherwise,
returns false.
  {
    return (rear == null);
  }

  public boolean isFull()
  // Returns false - a linked queue is never full.
  {
    return false;
  }
```

```
  public CircularLinkedQueue()
  {
    //front = null;
    rear = null;
  }

  public void enqueue(T element)
  // Adds element to the rear of this queue.
  {
    LLNode<T> newNode = new LLNode<T>(element);
    if (rear == null)
    {
      rear = newNode;
      rear.setLink(rear);
    } else {
      newNode.setLink(rear.getLink());
      rear.setLink(newNode);
      rear = newNode;
    }
    numElements++;
  }

  public T dequeue()
  // Throws QueueUnderflowException if this queue is
empty;
  // otherwise, removes front element from this queue
and returns it.
  {
    if (isEmpty())
      throw new QueueUnderflowException("Dequeue
attempted on empty queue.");
    else
    {
      T element;
      element = rear.getLink().getInfo();
      if (rear == rear.getLink())
        rear=null;
      else {
        rear.setLink( rear.getLink().getLink() );
      }
      //if (front == null)
      //  rear = null;
      numElements--;
      return element;
    }
  }
```

Here is a nice overview of our Queue Architecture, from the Chapter 04 PPT, pg 66: