1. Based on what we know about linked lists, stacks, and queues, design a linked queue (a queue using a linked-list to store the data in the structure)
2. Design, implement, and test a Queue data structure that:
   1. uses a **linked-list** to store values in the queue
   2. has an **enqueue** method that will appropriately **add** a value to the **back** of the queue as an appropriate element

A screen shot of a computer program

Description automatically generated with low confidence

* 1. has a **dequeue** method that will appropriately **remove** an element from the **front** of the queue **and return its value**

A picture containing text, screenshot, font, software

Description automatically generated

* 1. Optionally has a **peek** method that **returns the value at the front** of the queue **without removing it**

A picture containing text, screenshot, font

Description automatically generated  
**Bonus** if you also create an array based Queue!

1. Analyze the complexity of your implementations (at least the run-time of the add, remove, and peek methods).  
   (**Note** that we will often consider operations not having to do with the structure as O(1), even if they might be expensive operations in terms of real-time or space used)  
   (**Note** that if you are not in class when we talk about Asymptotic Big-O notation, you can find tons of good examples online)

All the functions I implemented were O(1) because instead of iterating throught the linked list to get to the values, I made a front and back pointer that always updated to the proper value after any given operation, making it take up slightly more storage but greatly increasing the runtime efficiency which would have been O(N) if I iterated through.

1. Tests: Be sure to include at least one test for each piece of functionality that should verify that your code is working!

A screen shot of a computer program

Description automatically generated with medium confidenceA picture containing text, screenshot

Description automatically generated

1. Be sure to commit changes regularly to your git repo
2. Once you have implemented and tested your code, add to the README file what line(s) of code or inputs and outputs show your work meeting each of the above requirements (or better, include a small screen snip of where it meets the requirement!).
3. Remember to submit a link to this project in Moodle to remind us to grade it!