

- 1 Explain at least one benefit and one drawback for using a linked list in place of a traditional array in the **Stack** and **Queue** data structures. (1)
- 2 Implement the `trimN()` method in our **LinkedList** class. This method should take **N** as a parameter and remove the last **N** nodes from the list. (1)
- 3 Implement **Queue** using a linked list. Ensure you maintain the use of generics as appropriate. (2)
- 4 Implement the `find()` method in our **LinkedList** class. This method should take some data, **key**, as a parameter and return the index where **key** exists as the data for a particular **Node**. This method should return the value `-1` if **key** does not exist in the linked list. (2)
- 5 *Doubly-Linked List*. Make all necessary changes to our **LinkedList** in order to implement a *doubly-linked list* in which each node has both **Next** and **Previous** attributes. In particular, ensure that `add()` and `remove()` work as intended. (4)
- 6 *Deque*. A double-ended queue or *deque* (pronounced "deck") is like a stack or a queue but supports adding and removing items from both ends. A deque stores a collection of items and supports the following operations: (6)

<code>isEmpty()</code>	is the deque empty?
<code>size()</code>	how many items are in the deque?
<code>pushLeft(item)</code>	add an item to the left end
<code>pushRight(item)</code>	add an item to the right end
<code>popLeft()</code>	remove and return an item from the left end
<code>popRight()</code>	remove and return an item from the right end

Implement a *deque* in Java using a doubly-linked list.