1. Use a linked list internally to implement a Stack
2. Use a linked list internally to implement a Queue

Q1

A stack has a “Last In, First Out” property. Pushing and popping only happen at the end of the stack. The pushing and popping functions would be simple. Locate the tail at first, and then connect the new node with the tail, or disconnect the current tail with the list. Pushing has a time complexity of O(1), as we just need to adjust the pointers near the tail; Popping also has a time complexity of O(1), as we just need to adjust the pointers near the tail; GetTop has a time complexity of O(1), as we just need to obtain the first item, without the need to use a loop to iterate over the stack; IsEmpty has a time complexity of O(1), as the comparison of the length with 0 is all that is needed

Q2

A queue has a “First In, First Out” property. Enqueuing happens at the end of the queue, while dequeuing happens at the top of the queue. The enqueuing function would locate the tail at first, and then connect the new node with the tail, while the dequeuing function would locate the head, and then disconnect the current head with the list. Enqueuing has a time complexity of O(1), as we just need to adjust the pointers near the tail; Dequeuing also has a time complexity of O(1), as we just need to adjust the pointers near the head; Size has a time complexity of O(1), as the size is maintained within the linked list class; IsEmpty has a time complexity of O(1), as the comparison of the length with 0 is all that is needed.