Assignment4 Report

In this assignment, we will learn how to use the queue data structure in C++. Overall, we'll use a linked list approach to implement the Queue. Each element in the Queue will be represented by a Node, containing the element's value and pointers to the previous and next nodes. First, we'll implement the Node class, which represents each element in the Queue. Each Node contains the element's value and pointers to the previous and next nodes. Then, we implement the Queue class. It maintains pointers to the head and tail of the Queue and provides methods for enqueue, dequeue, checking the size, and printing the Queue from head to tail. In enqueue, I create a new Node with the provided element value. and then we need to make sure that the queue is empty or not. If the queue is empty (head and tail are both NULL), we set both head and tail to point to the new Node. Otherwise, we update the pointers to maintain the queue's integrity: the current tail's next pointer points to the new Node, and the new Node's previous pointer points to the current tail. Finally, we update the tail pointer to

the new Node, effectively adding it to the rear end of the queue. In dequeue, we first check if the queue is empty by verifying if the head pointer is NULL. If it is, there are no elements to dequeue, so we return -1. If there is only one element in the queue (head->next is head), we reset both head and tail pointers to NULL, effectively emptying the queue. Otherwise, we update the pointers to skip the current head Node: the tail's next pointer points to the current head's next Node, and the next Node's previous pointer points to the tail. Then, we update the head pointer to the next Node. Finally, we delete the current head Node to free up memory.