Programming Assignment #3
Due by March 5 at 5pm to eCampus

First Name: Asa Last Name: Hayes UIN: 525003952

User Name AsaHayes E-mail address: asahayes@tamu.edu

Please list all sources in the table below including web pages which you used to solve or implement the current homework. If you fail to cite sources you can get a lower number of points or even zero, read more: Aggie Honor System Office

Type of sources	
People	
Web pages (provide URL)	
Other Sources	Several programming assignments from CSCE-121 with Dr. Moore

I certify that I have listed all the sources that I used to develop the solutions/codes to the submitted work.

"On my honor as an Aggie, I have neither given nor received any unauthorized help on this academic work."

Your Name: Asa R. Hayes   Date: 11 March, 2018

<p style="text-align: center">Report</p>

<p style="text-align: center">April 11, 2018</p>

Part I

Program Description and Purpose of Assignment

The first part of the assignment is an implementation of a Doubly Linked List data structure. After the initial implementation using <int> type as values for the nodes was a generic implement allowing for any value type to be used for a node. The second part of the assignment builds onto the doubly linked list created in the first part to create a MinQueue structure.

The purpose of this assignment is to give experience in creating linked lists, to illustrate the similarity between the Linked List and Queue data structures, and to introduce the use of ADPs (Adapter Design Patterns).

Part II

Description of Data Structures

1 Part 1: Doubly Linked List and Generic Doubly Linked List

The first part of the assignment is an implementation of a Doubly Linked List data structure. A Linked List is a list of nodes, where each node contains a value/objects as well as a pointer to the next node, terminated on each end by a null node. A doubly linked list gives each node a pointer to the previous node as well, allowing for backwards traversal. After the initial implementation using <int> type as values for the nodes was a generic implement allowing for any value type to be used for a node.

Functions with runtime of O(1): Default constructor, insertFirst(), insertLast(), removeFirst(), removeLast(), first(), last(), getFirst(), getAfterLast(), isEmpty()

Functions with runtime of O(n): Copy Assignment and Operator, insertBefore(), insertAfter(), removeBefore(), removeAfter(), doublyLinkedListLength

2 Part 2: Doubly Linked List-based MinQueue

The second part of the assignment builds onto the doubly linked list created in the first part to create a queue. A queue is a list of elements where elements can only be inserted/pushed/enqueued into the end of the list and only be accessed/popped/dequeued from the front of the list. The MinQueue class is essentially overlaid onto a doubly linked list, giving it a few more rules to follow. The only accessor founctions available to MinQueue are enqueue(), which uses DoublyLinkedList's insertLast(), and dequeue, which uses DoublyLinkedList's removeFront(). In addition, a new function min() traverses the list to find the smallest element. The size of the queue is handled by a member that is incremented/decremented on every enqueue/dequeue and isEmpty() uses the same function from DoublyLinkedList.

Functions with runtime of O(1): Constructor, isEmpty, first(), enqueue(int x), dequeue(), size()

Functions with runtime of O(n): min(), deconstructor
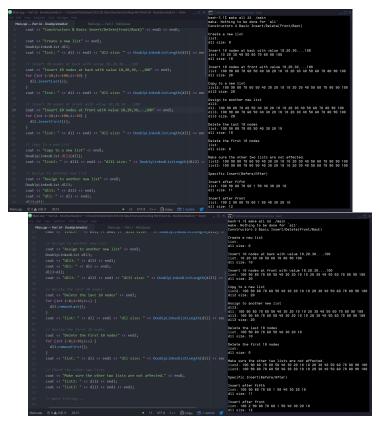
Part III
Instructions for Compilation

Use "make all && ./Main" work for the basic DoublyLinkedList and MinQueue, Use "make all && ./TemplatedMain" for the templated DoublyLinkedList. All files are already in order within their directories.
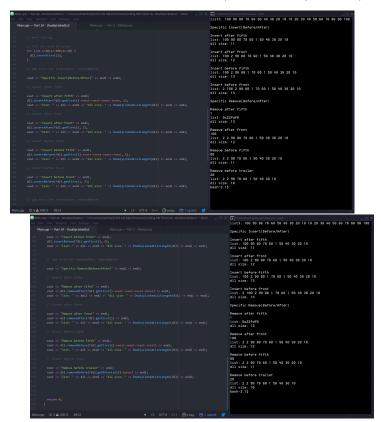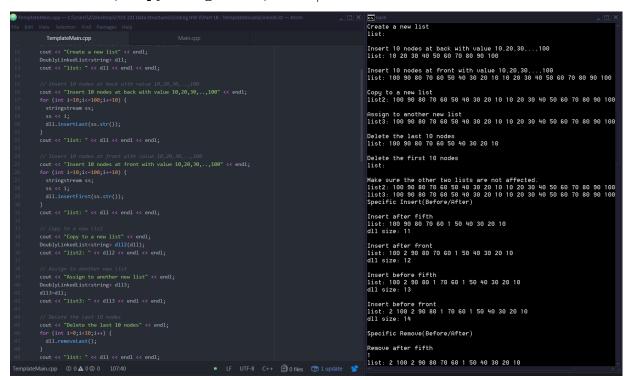
Part IV
Evidences of Testing

3 Evidence: Part 1A (DoublyLinkedList<int>)

3.1 Test 1: Initialization, Front/Back Insertion, Copy Constructor, Copy Assignment, Front/Back Deletion
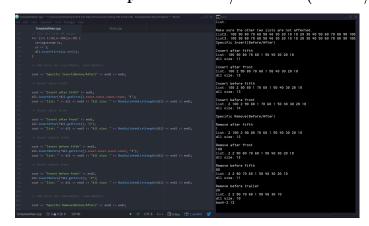
3.2 Test 2: Specific Insert/Remove (Before/After)

4 Evidence: Part 1B (generic DoublyLinkedList<string>)

4.1 Test 1: Initialization, Front/Back Insertion, Copy Constructor, Copy Assignment, Front/Back Deletion

4.2 Test 2: Specific Insert/Remove (Before/After)

5 Evidence: Part 2 (minQueue)

5.1 Test 1: All Functions