CSCE 221 Assignment 3 Cover Page

First Name Hunter Last Name Cleary UIN 625001547

User Name hncleary E-mail address hncleary@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 on Aggie Honor System Office website: `http://aggiehonor.tamu.edu/`

Type of sources				
People				
Web pages (provide URL)	Listed Below			
Printed material	Data Structures and Algorithms (Textbook)			
Other Sources				

Websites
https://en.wikipedia.org/wiki/Doubly_linked_list
https://www.geeksforgeeks.org/doubly-linked-list/
https://stackoverflow.com/questions/32938119/doubly-linked-list-insert-before-function-c

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 Hunter Cleary Date 3-8-18

1. The description of an assignment problem.

   The assignment required that a doubly linked list classes be created and implemented in C++. The first class was created for integer data types. A second, templated, class was created for data of arbitrary type. A MinQueue data structure was also created that supports queue operations and a min() function.

2. The description of data structures and algorithms used to solve the problem.

   (a) Provide definitions of data structures by using Abstract Data Types (ADTs)

   A doubly linked list is a set of nodes that each hold two pointers. The pointers of each node point to the previous node and the next node. Typically there also exists a search node which is able to traverse the list. The pointers between data nodes allow the search node to move in both directions. Doubly linked lists allow for optimized insertion, deletion, and access at any node.

   (b) Write about the ADTs implementation in C++.

   The DoublyLinkedList class in C++ contains a set of functions that work with the DListNode Structure. DListNode contains the stored data and the pointers to the previous and proceeding node. The DoublyLinkedList Class contains functions for various forms of output, insertion, and removal. It also contains a copy constructor for moving nodes into a new list and destrcutor for deleting the items.

   (c) Describe algorithms used to solve the problem.

   Linear search was used several times to find desired nodes in order to insert / delete at a specific point in the list. A function was created to find minimum values, which iterates through the list and compares at each node.

   (d) Analyze the algorithms according to assignment requirements.
   Doubly Linked List
   insertFirst - O(1) - Inserts a node at the beginning of the list. Creates a new node and then reassigns the pointer.
   insertLast - O(1) - Functions the same way as insertFirst, but at the end of the list.
   removeFirst - O(1) - Deletes the first node and reassigns the pointer of the next node.
   removeLast - O(1) - Deletes the last node and reassigns the pointer of the previous node.
   first(),last() - O(1) - Checks the value of the first / last node.
   isEmpty - O(1) - Makes a comparison to see if the list is NULL.
   Copy Constructor - O(n) - The function copies every item and places it into another list.
   Output Operator - O(n) - Prints out each node in the list.
   ListLength - O(n) - Iterates through the list and counts the number of items present.
   insertAfter, insertBefore - O(n) - Searches through the list for a specific node and then places it before / after. Reassigns corresponding pointers.
   removeBefore, removeAfter - O(n) - Searches through the list for a specific node and then deletes the node before / after it. Reassigns pointers to compensate for missing node.
   MinQueue
   enqueue - O(1) - Inserts node at the start of the queue.
   dequeue - O(1) - Removes node from the end of the queue.
   isEmpty - O(1) - Makes a comparison to see if the list is NULL.
   min - O(n^2) - Looks at each value of the list, makes copmarisons and the outputs the smallest value in the

list.

3. A C++ organization and implementation of the problem solution

   (a) Provide a list and description of classes or interfaces used by a program such as classes used to implement the data structures or exceptions.

   DoublyLinkedList - class for doubly linked list containing integer values
   TemplateLinkedList - class for doubly linked list containing generic data types
   MinQueue - wrapper for doubly linked list that enables queue functions

   (b) Include in the report the class declarations from a header file (.h) and their implementation from a source file (.cpp).

```
#include <cstdlib>
#include <iostream>
using namespace std;
class DoublyLinkedList; // class declaration

// list node
struct DListNode {
  int obj;
  DListNode *prev, *next;
  DListNode(int e=0, DListNode *p = NULL, DListNode *n = NULL)
    : obj(e), prev(p), next(n) {}
  int getElem() const { return obj; }
  DListNode * getNext() const { return next; }
  DListNode * getPrev() const { return prev; }
};

// doubly linked list
class DoublyLinkedList {
private:
  DListNode header, trailer;
public:
  DoublyLinkedList() : header(0), trailer(0) // constructor
  { header.next = &trailer; trailer.prev = &header; }
  DoublyLinkedList(const DoublyLinkedList& dll); // copy constructor
  ~DoublyLinkedList(); // destructor
  DoublyLinkedList& operator=(const DoublyLinkedList& dll); // assignment operator
  // return the pointer to the first node
  DListNode *getFirst() const { return header.next; }
  // return the pointer to the trailer
  const DListNode *getAfterLast() const { return &trailer; }
  // return if the list is empty
  bool isEmpty() const { return header.next == &trailer; }
  int first() const; // return the first object
  int last() const; // return the last object
  void insertFirst(int newobj); // insert to the first of the list
  int removeFirst(); // remove the first node
```

```
  void insertLast(int newobj); // insert to the last of the list
  int removeLast(); // remove the last node
  void insertAfter(DListNode &p, int newobj); // insert after desired node
  void insertBefore(DListNode &p, int newobj); // insert before selected node
  int removeAfter(DListNode &p); // removes node after selected
  int removeBefore(DListNode &p); //removes node before selected
};

// output operator
ostream& operator<<(ostream& out, const DoublyLinkedList& dll);
// return the list length
int DoublyLinkedListLength(DoublyLinkedList& dll);
```

(c) Provide features of the C++ programming paradigms like Inheritance or Polymorphism in case of object oriented programming, or Templates in the case of generic programming used in your implementation.

TemplateDoublyLinkedList is templated version of the DoublyLinkedList class. The new class allows for any input of generic type T. The original class was altered using template<class T> to allow for the generic input.

4. A user guide description how to navigate your program with the instructions how to:

(a) compile the program: specify the directory and file names, etc.

Each directory (DoublyLinkedList, TemplateDoublyLinkedList, and MinQueue) has a make file.

(b) run the program: specify the name of an executable file.

Once compiled, the programs should run with: ./Main, ./TemplatedMain, and ./MinQueue, respectively.

5. Specifications and description of input and output formats and files

All files run solely in the command line once compiled.

6. Provide types of exceptions and their purpose in your program.

(a) logical exceptions (such as deletion of an item from an empty container, etc.)

The program avoids using nodes that are NULL. There are several safeguards implemented so that memory is properly handled.

7. Test your program for correctness using valid, invalid, and random inputs (e.g., insertion of an item at the beginning, at the end, or at a random place into a sorted vector). Include evidence of your testing, such as an output file or screen shots with an input and the corresponding output.

compute.cse.tamu.edu - PuTTY

```
)
:: ./Main
Create a new list
list:

Insert 10 nodes at back with value 10,20,30,..,100
list: 10 20 30 40 50 60 70 80 90 100

Insert 10 nodes at front with value 10,20,30,..,100
list: 100 90 80 70 60 50 40 30 20 10 10 20 30 40 50 60 70 80 90 100

Copy to a new list
list2: 100 90 80 70 60 50 40 30 20 10 10 20 30 40 50 60 70 80 90 100

Assign to another new list
list3: 100 90 80 70 60 50 40 30 20 10 10 20 30 40 50 60 70 80 90 100

Delete the last 10 nodes
list: 100 90 80 70 60 50 40 30 20 10

Delete the first 10 nodes
list:

Make sure the other two lists are not affected.
list2: 100 90 80 70 60 50 40 30 20 10 10 20 30 40 50 60 70 80 90 100
list3: 100 90 80 70 60 50 40 30 20 10 10 20 30 40 50 60 70 80 90 100

Insert After Test
list2: 100 99999 90 80 70 60 50 40 30 20 10 10 20 30 40 50 60 70 80 90 100

Remove After Test
Contents of node removed: 99999
list2: 100 90 80 70 60 50 40 30 20 10 10 20 30 40 50 60 70 80 90 100

Insert Before Test
list2: 9898989 100 90 80 70 60 50 40 30 20 10 10 20 30 40 50 60 70 80 90 100

Remove Before Test
Contents of node removed: 9898989
list2: 100 90 80 70 60 50 40 30 20 10 10 20 30 40 50 60 70 80 90 100

Length 1: 0
Length 2: 20
Length 3: 20
sh: pause: command not found

[hncleary]@compute ~/ProgrammingAssignment3/DoublyLinkedList> (21:40:38 03/08/18
)
::
```

compute.cse.tamu.edu - PuTTY

```
[hncleary]@compute ~/ProgrammingAssignment3/TemplateDoublyLinkedList> (21:41:42 03/08/18)
:: ./TemplateMain
Create a new list
list:

Insert 10 nodes at back with value 10,20,30,..,100
list: 10 20 30 40 50 60 70 80 90 100

Insert 10 nodes at front with value 10,20,30,..,100
list: 100 90 80 70 60 50 40 30 20 10 10 20 30 40 50 60 70 80 90 100

Copy to a new list
list2: 100 90 80 70 60 50 40 30 20 10 10 20 30 40 50 60 70 80 90 100

Assign to another new list
list3: 100 90 80 70 60 50 40 30 20 10 10 20 30 40 50 60 70 80 90 100

Delete the last 10 nodes
list: 100 90 80 70 60 50 40 30 20 10

Delete the first 10 nodes
list:

Make sure the other two lists are not affected.
list2: 100 90 80 70 60 50 40 30 20 10 10 20 30 40 50 60 70 80 90 100
list3: 100 90 80 70 60 50 40 30 20 10 10 20 30 40 50 60 70 80 90 100

Insert After Test
list2: 100 howdy 90 80 70 60 50 40 30 20 10 10 20 30 40 50 60 70 80 90 100

Remove After Test
Contents of node removed: howdy
list2: 100 90 80 70 60 50 40 30 20 10 10 20 30 40 50 60 70 80 90 100

Insert Before Test
list2: howdy 100 90 80 70 60 50 40 30 20 10 10 20 30 40 50 60 70 80 90 100

Remove Before Test
Contents of node removed: howdy
list2: 100 90 80 70 60 50 40 30 20 10 10 20 30 40 50 60 70 80 90 100

Length 1: 0
Length 2: 20
Length 3: 20
sh: pause: command not found

[hncleary]@compute ~/ProgrammingAssignment3/TemplateDoublyLinkedList> (21:41:47 03/08/18)
::
```

```
[hncleary]@compute ~/ProgrammingAssignment3/MinQueue> (21:42:44 03/08/18)
:: ./MinQueue
Create a new list
list:

Insert 10 nodes at back with value 10,20,30,..,100
list: 10 20 30 40 50 60 70 80 90 100

Insert 10 nodes at front with value 10,20,30,..,100
list: 100 90 80 70 60 50 40 30 20 10 10 20 30 40 50 60 70 80 90 100

Testing Enqueue Function:
list: 15 14 13 12 100 90 80 70 60 50 40 30 20 10 10 20 30 40 50 60 70 80 90 100

Testing Dequeue Function:
list: 15 14 13 12 100 90 80 70 60 50 40 30 20 10 10 20 30 40 50 60 70 80 90

Testing Size Function:
Size: 23

Testing isEmptyFunction:
Is the list empty?:  0

Testing Min Function:
The minimum value of this list is: 10

sh: pause: command not found

[hncleary]@compute ~/ProgrammingAssignment3/MinQueue> (21:42:49 03/08/18)
::
```