CSCE 221 Assignment 3 Cover Page

First Name McKenzie Last Name Burch UIN 225005240

User Name mburch13gig-em E-mail address mburch13gig-em@gmail.com

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)	www.geeksforgeeks.org www.cplusplus.com	stackoverflow.com		
Printed material	Textbook			
Other Sources	Lecture Slides Personal Notes			

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 McKenzie Burch Date July 27, 2018

CSCE 221 Assignment 3

Summer 2018

due to eCampus by July 27, and demonstration of Part 1 in labs on July 23

Objective

This is an individual assignment which has two parts.

1. Part 1: C++ implementation of `DoublyLinkedList` for `int` and generic types based on the provided supplementary code.
2. Part 2: C++ implementation of `MinQueue` data structure that can store *comparable elements.*

Part 1: Implementation of DoublyLinkedList

1. Untar supplementary code `221-A3-code.tar`. Use the 7-zip software to extract the files in Windows, or use the following command in Linux.

   ```
   tar xfv 221-A3-code.tar
   ```

2. `DoublyLinkedList` for integers
   (a) Most of the code is extracted from the lecture slides. An exception structure is defined to complete the program.
   (b) You need to complete the functions which are declared in the header file `DoublyLinkedList.h`.
   (c) Type the following commands to compile the program

   ```
   make
   ```

   (d) The main program includes examples of creating doubly linked lists, and demonstrates how to use them. Type the following command to run the executable file:

   ```
   ./run-dll
   ```

   (e) Test the doubly linked list functions in `main`.
3. Implement a templated version of the class `DoublyLinkedList` and test the functions for correctness. Follow the instructions below:
   (a) Templates should be declared and defined in a header `.h` file. Move the content of `DoublyLinkedList.cpp` and `DoublyLinkedList.h` to `TemplateDoublyLinkedList.h`
   (b) Replace `int obj` by `T obj` in the `struct DListNode` so the list nodes store generic `T` objects instead of integers. Later on, when a `DListNode` object is created, say, in the main function, `T` can be specified as a `char`, `string` or a user-defined class.
   (c) To create a templated class with a generic type `T`, you must replace a declaration/return type `int` by `T` (except for the `count` variable).
      i. To use the generic type `T`, you must change each type declaration.
      ii. Use the generic type `T` anywhere throughout the class `TemplateLinkedList`.
   (d) Add the keyword `template <typename T>` before a class declaration.
   (e) In each member function signature, replace `DoublyLinkedList::` by `DoublyLinkedList<T>::`
   (f) If a member function is defined outside the class declaration, change the function signature, that is, replace `LinkedList::` by `TemplateLinkedList<T>::`

(g) To use the generic type `T` anywhere throughout the class `DListNode` and `DoublyLinkedList`, you must declare (add) `template <typename T>` before classes and member functions defined outside the class declaration.

4. Compile and run the generic version in a similar way as for `int` type. Type the following commands to compile the program.

   ```
   make
   ```

5. The main program includes examples of creating doubly linked lists of `string`, and demonstrates how to use them. Type the following command to run the executable.

   ```
   ./run-tdll
   ```

Part 2: Implementation of MinQueue data structure based on DoublyLinkedList

The `MinQueue` data structure should store *the comparable elements* that support the queue operations: `enqueue(x)`, `dequeue()`, `size()`, `isEmpty()`, and in addition the `min()` operation that returns (but not deletes) the smallest value currently stored in the queue.

Use the adapter design pattern for implementation of `MinQueue` that work together with the class `DoublyLinkedLists` defined in the Part 1. The runtime worst case of all operations except `min()` should be *constant,* $O(1)$.

The implementation details of the `MinQueue` operations, justification of their running time, and tests for correctness should be provided in the part 2 of the report.

1. enqueue(x): inserts element into the queue
   - Runtime: $O(1)$
2. dequeue(): removes element from queue
   - Runtime: $O(1)$
3. size(): returns size of queue
   - Runtime: $O(1)$
4. isEmpty(): returns boolean for if queue is empty
   - Runtime: $O(1)$
5. min(): returns smallest value currently stored in queue
   - Runtime: $O(1)$

```
#include <stdio.h>
#include "DoublyLinkedList"

class MinQueue(){
private:
    DoublyLinkedList dll;
public:
    MinQueue() : dll() {}   //constructor
    ~MinQueue() {dll.~DoublyLinkedList(); } //destrucot
    void enqueue(int x) {ll.insertLast(x);}
    void dequeue() throw(QueueEmptyException);
    int size() const { return count; }
    bool isEmpty() const {ll.insertLast(elem)};
    int min();
};

void MinQueue::dequeue() throw(QueueEmptyException){
    if( isEmpty() )
        throw QueueEmptyException("Access to an empty queue");
    return ll.removeFirst();
}

int MinQueue::min(){
    int min;
    DListNode *temp;
    while(head.next != NULL){
        min = temp->elem;
        if(temp->next->elem < min){
            min = temp->next->elem;
        }
    }
    return min;
}
```

What to submit to eCampus?

- Create a directory for the Part 1 that includes: `DoublyLinkedList` source code for `int` and generic types, typed report with description of the linked list implementation, complexity analysis of code expressed in terms of big-O, and the test cases done for correctness.
- Create a directory for the Part 2 that includes: `Min`Queue source code, typed report with description of the `Min`Queue class implementation, complexity analysis of code expressed in terms of big-O, and the test cases done for correctness.
- Make a tar file that contains the Part 1 and Part 2 directories and submit it to eCampus for grading.