**CSC 2200 – Computer Science II**

**Lab #07**

**Take-Home**

**60 Points**

1. Save a copy of this document with your name and the assignment number somewhere in the file name. For example, the file name *“Jane\_Doe\_CSC2200\_Lab1.docx”*
2. Copy-and-paste your answers (e.g., C++ source code) into the document.
3. Copy-and-paste the program output window.
4. Submit the following files separately (do not compress the files) to the Blackboard in one submission:

1) This document as a word document (i.e., with the extension ***.doc*** or ***.docx***).

2) All C++ source code solution file(s) (only the ***.cpp*** and ***.h*** files) to the Canvas item associated with this assignment/lab solution. ***\*\*If you modified it, submit it***

\*Submit entire Visual Studio solution, if possible, otherwise only the required files.

Questions:  
- Implement the Queue ADT (40 points) using the linked list approach  
- Programming Exercise 3 (20 points)

\* **Copying-and-pasting your *C++ program code* to a Word document**

1) From within the Visual Studio program, press **CTRL-A** and press **CTRL-C**.

2) From within the Word document, press **CTRL-V**.

#include "QueueLinked.h"

template <typename DataType>

QueueLinked<DataType>::QueueNode::QueueNode(const DataType& nodeData, QueueNode\* nextPtr)

: dataItem(nodeData), next(nextPtr)

{

}

template <typename DataType>

QueueLinked<DataType>::QueueLinked(int maxNumber)

: front(nullptr), back(nullptr)

{

}

template <typename DataType>

QueueLinked<DataType>::QueueLinked(const QueueLinked& other)

: front(nullptr), back(nullptr)

{

QueueNode\* otherCurrent = other.front;

while (otherCurrent != nullptr) {

enqueue(otherCurrent->dataItem);

otherCurrent = otherCurrent->next;

}

}

template <typename DataType>

QueueLinked<DataType>& QueueLinked<DataType>::operator=(const QueueLinked& other)

{

if (this != &other) {

clear();

QueueNode\* otherCurrent = other.front;

while (otherCurrent != nullptr) {

enqueue(otherCurrent->dataItem);

otherCurrent = otherCurrent->next;

}

}

return \*this;

}

template <typename DataType>

QueueLinked<DataType>::~QueueLinked()

{

clear();

}

template <typename DataType>

void QueueLinked<DataType>::enqueue(const DataType& newDataItem) throw (logic\_error)

{

QueueNode\* newNode = new QueueNode(newDataItem, nullptr);

if (isEmpty()) {

front = newNode;

back = newNode;

}

else {

back->next = newNode;

back = newNode;

}

}

template <typename DataType>

DataType QueueLinked<DataType>::dequeue() throw (logic\_error)

{

if (isEmpty()) {

throw logic\_error("Queue is empty");

}

DataType temp = front->dataItem;

QueueNode\* tempPtr = front;

front = front->next;

if (front == nullptr) {

back = nullptr;

}

delete tempPtr;

return temp;

}

template <typename DataType>

void QueueLinked<DataType>::clear()

{

while (!isEmpty()) {

dequeue();

}

}

template <typename DataType>

bool QueueLinked<DataType>::isEmpty() const

{

return front == nullptr;

}

template <typename DataType>

bool QueueLinked<DataType>::isFull() const

{

return false;

}

template <typename DataType>

void QueueLinked<DataType>::putFront(const DataType& newDataItem) throw (logic\_error)

{

QueueNode\* newNode = new QueueNode(newDataItem, front);

if (isEmpty()) {

back = newNode;

}

front = newNode;

}

template <typename DataType>

DataType QueueLinked<DataType>::getRear() throw (logic\_error)

{

if (isEmpty()) {

throw logic\_error("Queue is empty");

}

return back->dataItem;

}

template <typename DataType>

int QueueLinked<DataType>::getLength() const

{

int length = 0;

QueueNode\* current = front;

while (current != nullptr) {

length++;

current = current->next;

}

return length;

}

template <typename DataType>

void QueueLinked<DataType>::showStructure() const

{

QueueNode\* p = front;

if (isEmpty()) {

cout << "Empty queue" << endl;

}

else {

cout << "Front\t";

while (p != nullptr) {

if (p == front) {

cout << "[" << p->dataItem << "] ";

}

else {

cout << p->dataItem << " ";

}

p = p->next;

}

cout << "\trear" << endl;

}

}

\*\* **Copying-and-pasting a C++ “*output window”* to a Word document**

1) From the Visual Studio output window, press **ALT-PrintScreen**.

2) From within the Word document, press **CTRL-V**.

A black screen with a black background

Description automatically generated