**CSC 2200 – Computer Science II**

**Lab #07**

**In-Class**

**40 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 array – based approach

\* **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 "QueueArray.h"

template <typename DataType>

QueueArray<DataType>::QueueArray(int maxNumber) : maxSize(maxNumber), front(-1), back(-1) {

dataItems = new DataType[maxSize];

}

template <typename DataType>

QueueArray<DataType>::QueueArray(const QueueArray& other) : maxSize(other.maxSize), front(other.front), back(other.back) {

dataItems = new DataType[maxSize];

for (int i = 0; i < maxSize; ++i) {

dataItems[i] = other.dataItems[i];

}

}

template <typename DataType>

QueueArray<DataType>& QueueArray<DataType>::operator=(const QueueArray& other) {

if (this != &other) {

delete[] dataItems;

maxSize = other.maxSize;

front = other.front;

back = other.back;

dataItems = new DataType[maxSize];

for (int i = 0; i < maxSize; ++i) {

dataItems[i] = other.dataItems[i];

}

}

return \*this;

}

template <typename DataType>

QueueArray<DataType>::~QueueArray() {

delete[] dataItems;

}

template <typename DataType>

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

if (isFull()) {

throw logic\_error("Queue is full. Cannot enqueue.");

}

//If the queue is empty, set front to 0

if (isEmpty()) {

front = 0;

}

//Move the back index circularly

back = (back + 1) % maxSize;

dataItems[back] = newDataItem;

}

template <typename DataType>

DataType QueueArray<DataType>::dequeue() throw (logic\_error) {

//Implement the dequeue operation here and return the removed item

if (isEmpty()) {

throw logic\_error("Queue is empty. Cannot dequeue.");

}

DataType removedItem = dataItems[front];

//If there is only one item in the queue, reset front and back to -1

if (front == back) {

front = -1;

back = -1;

}

else {

//Move the front index circularly

front = (front + 1) % maxSize;

}

return removedItem;

}

template <typename DataType>

void QueueArray<DataType>::clear() {

//Implement the clear operation here

front = -1;

back = -1;

}

template <typename DataType>

bool QueueArray<DataType>::isEmpty() const {

//Implement the isEmpty operation here and return true if the queue is empty, false otherwise

return (front == -1);

}

template <typename DataType>

bool QueueArray<DataType>::isFull() const {

//Implement the isFull operation here and return true if the queue is full, false otherwise

return ((back + 1) % maxSize == front);

}

template <typename DataType>

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

if (isFull()) {

throw logic\_error("Queue is full. Cannot putFront.");

}

//If the queue is empty, set front and back to 0

if (isEmpty()) {

front = 0;

back = 0;

}

else {

// Move the front index circularly

front = (front - 1 + maxSize) % maxSize;

}

//Add the new item to the front

dataItems[front] = newDataItem;

}

template <typename DataType>

DataType QueueArray<DataType>::getRear() throw (logic\_error) {

//Implement the getRear operation here and return the rear item

if (isEmpty()) {

throw logic\_error("Queue is empty. Cannot getRear.");

}

return dataItems[back];

}

template <typename DataType>

int QueueArray<DataType>::getLength() const {

//Implement the getLength operation here and return the current length of the queue

return (front == -1) ? 0 : ((back - front + maxSize) % maxSize + 1);

}

//--------------------------------------------------------------------

template <typename DataType>

void QueueArray<DataType>::showStructure() const

// Array implementation. Outputs the data items in a queue. If the

// queue is empty, outputs "Empty queue". This operation is intended

// for testing and debugging purposes only.

{

int j; // Loop counter

if (front == -1)

cout << "Empty queue" << endl;

else

{

cout << "Front = " << front << " Back = " << back << endl;

for (j = 0; j < maxSize; j++)

cout << j << "\t";

cout << endl;

if (back >= front)

for (j = 0; j < maxSize; j++)

if ((j >= front) && (j <= back))

cout << dataItems[j] << "\t";

else

cout << " \t";

else

for (j = 0; j < maxSize; j++)

if ((j >= front) || (j <= back))

cout << dataItems[j] << "\t";

else

cout << " \t";

cout << 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**.



