WAYNE STATE UNIVERSITY
COLLEGE OF ENGINEERING

Computer Science Department

CSC 2201: Computer Science II – Lab
Lab7

Description:

You will implement a queue ADT using a singly linked list.

Goals:

Learn how to implement a queue ADT with linked data structures and arrays using C++ pointers.

Where to Start:

1. Download Lab7_Work.zip from Blackboard
2. Unzip the file
3. Open the solution in Microsoft Visual Studio
4. Make sure that the project does not show any compile errors.
5. Implement methods and operations of the QueueArray class:

5.1 Implement `QueueArray<DataType>::QueueArray(int maxNumber)`

```
Testing array implementation

Commands:
  H   : Help (displays this message)
  +x  : Enqueue x
  -   : Dequeue
  C   : Clear the queue
  E   : Empty queue?
  F   : Full queue?
  >x  : Put x at front     (  Active : Programming Exercise 2)
  =   : Get x from rear    (  Active : Programming Exercise 2)
  #   : Length             (  Active : Programming Exercise 3)
  Q   : Quit the test program

Empty queue
```

5.2 Implement `QueueArray<DataType>::QueueArray(const QueueArray& other)`

5.3 Implement `QueueArray<DataType>::~QueueArray()`

5.4 Implement `void QueueArray<DataType>::enqueue(const DataType& newDataItem) throw (logic_error)`

```
Command: +a
Enqueue a
Front = 0  Back = 0
0       1       2       3       4       5       6       7
a
```

5.5 Implement `DataType QueueArray<DataType>::dequeue() throw (logic_error)`
Destructor. Deallocates the memory used to store the nodes in the list.

```
Command: +c
Enqueue c
Front = 0  Back = 2
0       1       2       3       4       5       6       7
a       b       c

Command: -
Dequeued a
Front = 1  Back = 2
0       1       2       3       4       5       6       7
        b       c
```

5.6 Implement `void void QueueArray<DataType>::clear() throw (logic_error)`

```
Command: C
Clear the queue
Empty queue
```

5.7 Implement `bool QueueArray<DataType>::isEmpty() const`

```
Command: E
Queue is empty
Empty queue
```

5.8 Implement `bool QueueArray<DataType>::isFull() const`

```
Command: F
Queue is NOT full
Empty queue
```

Activate `LAB7_TEST2` and `LAB7_TEST3` in the config.h file.

5.9 Implement `void QueueArray<DataType>::putFront(const DataType& newDataItem) throw (logic_error)`

```
Command: +c
Enqueue c
Front = 0  Back = 2
0       1       2       3       4       5       6       7
a       b       c

Command: >x
Put x in front
Front = 7  Back = 2
0       1       2       3       4       5       6       7
a       b       c                                       x
```

5.10 Implement `DataType QueueArray<DataType>::getRear() throw (logic_error)`

```
Command: =
Got c from rear
Front = 7  Back = 1
0       1       2       3       4       5       6       7
a       b                                               x
```

5.11 Implement `int QueueArray<DataType>::getLength() const`

```
Command: #
Length = 3
Front = 7  Back = 1
0       1       2       3       4       5       6       7
a       b                                               x
```

6. Activate `LAB7_TEST1` in the config.h file to implement the queue ADT with linked list.
7. Implement the following methods in the class QueueLinked

   7.1 Implement `QueueLinked<DataType>::QueueNode::QueueNode(const DataType& nodeData, QueueNode* nextPtr)`

   7.2 Implement `QueueLinked<DataType>::QueueLinked(int maxNumber = Queue<DataType>::MAX_QUEUE_SIZE)`

```
Testing linked implementation

Commands:
  H   : Help (displays this message)
  +x  : Enqueue x
  -   : Dequeue
  C   : Clear the queue
  E   : Empty queue?
  F   : Full queue?
  >x  : Put x at front     (  Active : Programming Exercise 2)
  =   : Get x from rear    (  Active : Programming Exercise 2)
  #   : Length             (  Active : Programming Exercise 3)
  Q   : Quit the test program

Empty queue

Command:
```

   7.3 Implement `QueueLinked<DataType>::QueueLinked(const QueueLinked& other)`

   7.4 Implement `QueueLinked<DataType>::~QueueLinked()`

   7.5 Implement `void QueueLinked<DataType>::enqueue(const DataType& newDataItem) throw (logic_error)`

```
Command: +x
Enqueue x
Front   [x]     rear
```

7.6 Implement `DataType QueueLinked<DataType>::dequeue() throw (logic_error)`

```
Command: +g
Enqueue g
Front    [x] f g          rear

Command: -
Dequeued x
Front    [f] g    rear
```

7.7 Implement `void QueueLinked<DataType>::clear()`

```
Command: C
Clear the queue
Empty queue
```

7.8 Implement `bool QueueLinked<DataType>::isEmpty() const`

```
Command: E
Queue is empty
Empty queue
```

7.9 Implement `bool QueueLinked<DataType>::isFull() const`

```
Command: F
Queue is NOT full
Empty queue
```

Activate `LAB7_TEST2` and `LAB7_TEST3` in the config.h file.

7.10 Implement `bool void QueueLinked<DataType>::putFront(const DataType& newDataItem) throw (logic_error)`

```
Command: +h
Enqueue h
Front    [f] g h          rear

Command: >x
Put x in front
Front    [x] f g h        rear
```

7.11 Implement `DataType QueueLinked<DataType>::getRear() throw (logic_error)`

```
Command: >x
Put x in front
Front    [x] f g h        rear

Command: =
Got h from rear
Front    [x] f g          rear
```

7.12 Implement `int QueueLinked<DataType>::getLength() const`

```
Command: #
Length = 3
Front   [x] f g         rear
```

8. Test your implementation using the program in the file test7.cpp.

Create a Zip file of your solution:

1. Right click on your solution in Solution Explorer
2. Click on “Open Folder in File Explorer”
3. Go one level up in file explorer
4. Right click on your solution folder
5. Add it to archive by creating a zip file

Upload the zipped file on Blackboard:

1. Go to Blackboard
2. Click on this course (CSC 2201: Computer Science II Lab)
3. Go to the folder “Labs”
4. Click on the “Lab6_Work” assignment
5. Upload your zipped file