[CS302-Data Structures] Homework 5: Queues

Instructor: Kostas Alexis
Teaching Assistants: Shehryar Khattak, Mustafa Solmaz
Fall 2018 Semester

Section 1. Queue ADT – Overview wrt Provided Code

Data Items

The data items in a queue are of generic DataType.

Structure

The queue data items are linearly ordered from least recently added (the front) to most recently added (the rear/tail). Data items are inserted at the rear of the queue (enqueued) and are removed from the front of the queue (dequeued).

Operations

```
Queue (int maxNumber = MAX_QUEUE_SIZE)
```

Requirements

None

Results

Constructor. Creates an empty queue. Allocates enough memory for a queue containing maxNumber data items (if necessary).

```
Queue (const Queue& other)
```

Requirements

None

Results

Copy constructor. Initializes the queue to be equivalent to the other Queue object parameter.

```
Queue& operator= (const Queue& other)
```

Requirements

None

Results

Overloaded assignment operator. Sets the queue to be equivalent to the other Queue object parameter and returns a reference to the modified queue.

```
~Queue()
```

Requirements

None

Results

Destructor. Deallocates the memory used to store the queue.

```
void enqueue (const DataType& newDataItem) throw ( logic_error )
```

Requirements

Queue is not full.

Results

Inserts newDataItem onto the rear of the queue.

```
DataType dequeue() throw ( logic_error )
```

Requirements

Queue is not empty

Results

Removes the most recently added (front) data item from the queue and returns it.

```
void clear()
```

Requirements

None

Results

Removes all the data items in the queue.

```
bool isEmpty() const
```

Requirements

None

Results

Returns true if the queue is empty. Otherwise, returns false.

```
bool isFull () const
```

Requirements

None

Results

Returns true if the queue is full. Otherwise, returns false.

```
void showStructure() const
```

Requirements

None

Results

Outputs the data items in a queue. If the queue is empty, it outputs "Empty queue". Note that this operation is intended for testing/debugging purposes only. It only supports queue data items that are one of c++ predefined data types (int, char, etc) or other data structures with overridden ostream operator<<.

Section 2. Implementation Approaches

As in the class, we consider both

- Array-based implementations
- Linked-List implementations

For studying you are referred to the course slides.

Section 3. Compilation Directions wrt Provided Code

For Linked-List based Implementation: Edit config.h and change the value of LAB7_TEST1 to 0 to test the link-based implementation. (Re)compile test7.cpp.

For Linked-List based Implementation: Edit config.h and change the value of LAB7_TEST1 to 1 to test the link-based implementation. (Re)compile test7.cpp.

Select to perform this assignment either using an array-based implementation or a linked-list based implementation. You do not have to solve it with both.

Section 4. Testing

Test your implementation of the linked list Queue ADT using the program in the file test7.cpp.

Section 5. Programming Exercise 1

In this exercise, you use a queue to simulate the flow of customers through a check-out line in a store. In order to create this simulation, you must model both the passage of time and the flow of customers through the line. You can model time using a loop in which each pass corresponds to a set time interval – 1 minute, for example. You can model the flow of customers using a queue in which each data item corresponds to a customer in the line.

In order to complete the simulation, you need to know the rate at which customers join the line, as well as the rate at which they are served and leave the line. Suppose the check-out line has the following properties.

- One customer is served and leaves the line every minute (assuming there is at least one customer waiting to be served during that minute).
- Between zero and two customers join the line every minute, where there is a 50% chance that no customers arrive, a 25% chance that one customer arrives, and a 25% chance that two customers arrive.

You can simulate the flow of customers through the line during a time period that is n minutes long using the following algorithm.

```
    Initialize the queue to empty.
While the simulation is not done
    Increment simulated time by one minute
    If the queue is not empty, then remove the customer at the front of the queue
    Compute a random number k between 0 and 3
    If k is 1, then add one customer to the line
    If k is 2, then add two customers to the line
    Otherwise (if k is 0 or 3), do not add any customers to the line
    Update queue statistics
```

Calling the rand function is a simple way to generate pseudo-random numbers (one option through <cstdlib>). Generating random numbers does vary platform because of compiler and operating system differences.

Step 1: Using the program shell given in the file storesim.cs as a basis create a program in storesim.cpp that uses the Queue ADT to implement the model described above. Your program should update the following information during each simulated minute.

- The total number of customers served.
- The combined length of time these customers spent waiting in line.
- The maximum length of time any of these customers spent waiting in line

The data that is stored in the queue should contain everything that is necessary to 1) represent the customer and 2) compute the previous statistics. To compute how long a customer waited to be served, you need the difference in time from when the customer arrived to when the customer exited the queue. There is no additional information needed in the statistics, nor is there any customer-specific information that is used for our simple simulation. Therefore, we can represent the customer in the queue simply by using the simulated time the customer entered the queue.

Step 2: Use your program to simulate the flow of customers through the line and complete the table provided below. Note that the average wait is the combined waiting time divided by the total number of customers served.

Results Table 7-2 (customer check-out line simulation)			
Time (minutes)	**Total number of customers served**	**Average wait**	**Longest wait**
30			
60			
120			
480			