East West University
Department of Computer Science and Engineering
A/2, Jahurul Islam Avenue, Aftabnagar, Dhaka.

Topic: Queue

Course: CSE 207 Data Structures
Instructor: Ahmed Abdal Shafi Rasel

Objective: The objective of this lab assignment is to provide students with hands-on experience in implementing and understanding queue data structures. Students will learn how to perform basic queue operations, such as enqueue, dequeue, and peek, using both array-based and linked list-based implementations. They will analyze the time complexity of these operations and explore the advantages and limitations of each implementation.

Lab Activities:

A queue or FIFO (first in, first out) is an abstract data type that serves as a collection of elements, with two principal operations: enqueue, the process of adding an element to the collection. (The element is added from the rear side) and dequeue, the process of removing the first element that was added. (The element is removed from the front side). It can be implemented by using both an array and a linked list.

Queue is helpful in the following kinds of scenarios.

1) When a resource is shared among multiple consumers. Examples include CPU scheduling and Disk Scheduling.

2) When data is transferred asynchronously (data not necessarily received at the same rate as sent) between two processes. Examples include IO Buffers, pipes, file IO, etc.

Problems to Solve:

<table>
<tr><td>Task 1</td><td>Implement a Queue using an Array.
1. Include the following operations:
✓ enqueue(int item): Adds an item to the rear of the queue.
✓ dequeue(): Removes and returns the front item.
✓ peek(): Returns the front item without removing it.
✓ isEmpty(): Checks if the queue is empty.
✓ isFull(): Checks if the queue is full.
✓ display(): Displays the elements in the queue.
2. Write a main function to test all operations</td></tr>
<tr><td>Task 2</td><td>Implement a Queue class using a singly linked list.
1. Include the same operations as in Task 1.
2. Ensure the enqueue() and dequeue() operations run in O(1) time.</td></tr>
<tr><td>Task 3</td><td>Write a program that copies the content from one queue to another
<table><tr><td>Input Data</td><td>Output Data</td></tr><tr><td>Q1: 1 2 3 4 5</td><td>Q2: 1 2 3 4 5</td></tr></table>.</td></tr>
<tr><td>Task 4</td><td>Write a program that will take a queue of integers and delete all negative integers without changing the order of the remaining elements in the queue.
<table><tr><td>Input Data</td><td>Output Data</td></tr><tr><td>Q: 1 2 -3 4 - 5</td><td>Q: 1 2 4</td></tr></table>.</td></tr>
</table>

<table>
<tr><th colspan="2">Bonus Problems</th></tr>
<tr><td>1.</td><td>A railway station operates a ticket counter system with only two booths. Passengers arrive at different times and are served in a First-Come, First-Served (FIFO) manner. The system dynamically assigns passengers to the shorter queue to minimize waiting time. The next passenger in line is immediately served if a booth is free.

Simulation Details:

1. Passenger Arrivals:
✓ Passengers arrive at random intervals (1-5 seconds apart).
✓ Each passenger has a random service time (2-6 seconds).
2. Dynamic Queue Allocation:
✓ The system maintains two queues (one for each booth).
✓ A newly arriving passenger joins the shorter queue.
✓ If a booth is free at the time of arrival, the passenger gets immediate service without waiting in the queue.
3. Service Processing:
✓ The system tracks which booth is occupied and serves the next passenger when a booth becomes free.
✓ Passengers are served in order of arrival within their respective queues, but new arrivals dynamically pick the shorter queue.
4. Queue Status Updates:
✓ The queue status is displayed after each service.
✓ The system prints which booth is currently serving which passenger.
✓ The simulation continues until all passengers have been served.</td></tr>
<tr><td>2.</td><td>Simulate a traffic light system where three signals (Red, Green, Yellow) operate in a circular manner:

1. The signal switches every n seconds.
2. The system must cycle continuously (Red → Green → Yellow → Red).
3. Implement this using a circular queue.</td></tr>
</table>

- Bonus problems don't need to be submitted as part of the assignment. Students are given these problems so they can have some ideas on how to proceed with projects. Students can work on similar problems like these, expand them, add more functionalities, and start building course projects.