COMP 202: Data Structures and Algorithms
Lab work 3: Implementing Queue Data Structure

1 Purpose

To implement linear data structures.

In this exercise, you will implement queue data structure in C++.

2 Background

Linear data structures organize data elements in a linear manner, i.e. each data element has only unique successor. Array, stack, queue, linked lists etc. are some examples of linear data structures.

2.1 Queue

A *Queue* is a *First-In-First-Out (FIFO)* data structure where the first element inserted is the first element removed. Unlike stacks, insertions and deletions are made at different ends. New elements are added at one end, called the *rear*, and elements are removed from the other end, called the *front*.

Principal operations on a queue are *enqueue* (adding an element into a queue) and *dequeue* (removing an element from a queue).

3 Tasks

1. Implement Queue data structure using an array as well as a linked list. Following operations must be available:
   (a) enqueue(element): Adds an element into the queue
   (b) dequeue(): Removes an element from the queue
   (c) isEmpty(): Checks if the queue is empty
   (d) isFull(): Checks if the queue is full
   (e) front(): Gives the element at the front
   (f) back(): Gives the element at the rear

   Also, write a test program to check if the queue implementations work properly.

4 Lab work submission

Submit your work via Canvas **within 2 weeks**. Your submission must include the following:

1. Your code
2. A report containing
   (a) the output of your program, and
   (b) answers to the questions posed in the labsheet, if any.
3. Link to your Git repository must be in a comment. Your repository must be private. The user whom you must give access will be communicated during the lab session.