COMP 202: Data Structures and Algorithms
Lab work 2: Implementing Linear Data Structures - Stack and Queue

1 Purpose

To implement linear data structures.

In this exercise, you will implement stack and queue data structures using arrays.

You will also learn Git[1], a version control system. You will keep your code in `https://gitlab.com` or `https://github.com`.

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 Stack

A *stack* is a linear data structure where all insertions and deletions are restricted to one end, called the *top*. It is also known as a *Last-In-First-Out (LIFO)* list because the last element inserted into a stack is the first element removed.

Principal operations on a stack are *push* (adding an element into a stack) and *pop* (removing an element from a stack).

2.2 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).

[1] https://git-scm.com

3 Tasks

1. Implement a generic stack with the following operations:
   (a) push(element): Adds an element into the stack
   (b) pop(): Removes an element from the stack
   (c) isEmpty(): Checks if the stack is empty
   (d) isFull(): Checks if the stack is full
   (e) top(): Gives the element at the top

   Also, write a test program to check if the stack implementation works properly.

2. Implement a generic queue with the following operations:
   (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 implementation works properly.

4 Lab work submission

Submit your work via KU ELF[2] **within 2 weeks**. Your submission must include the following:

1. Source code
2. A report containing
   (a) the output of your program, and
   (b) answers to the questions posed in the labsheet, if any.

[2] `elf.ku.edu.np`