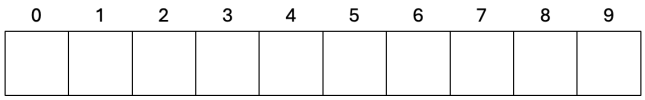
Course: Data Structures (CSE CS203A, 114-1)
Quiz II: Array, Linked List, Stack and Queue
October 21, 2025, 16:30~17:00

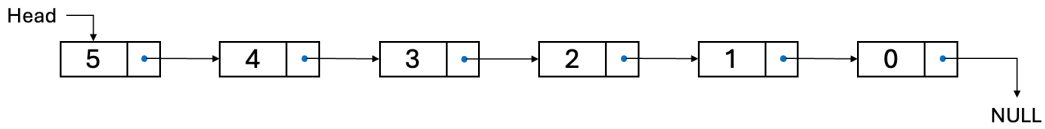
Student ID: Student Name:

Data Structures: Visualization

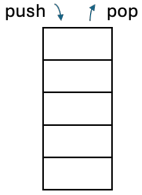
(1) Array



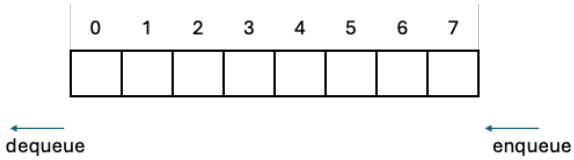
(2) Linked List



(3) Stack



(4) Queue



Q1: (30 pts; 10 pts for each) **Describe the mechanism of the function**

MoveTo(node *head, node *target, node*destination)

A1: Write a short paragraph explaining how the **MoveTo** function works (you may answer in English or Mandarin).

① Are there any **additional variables** required? If so, explain why they are necessary.

Variable	Purpose
prevTarget	Stores the node before the target to facilitate detachment
destination	Marks the node after which the target will be reinserted

The MoveTo() function **relocates a node** (target) to a new position after the node specified by destination within a singly linked list.
This requires **pointer manipulation**, not data swapping.

② **Draw a visualization** of the singly linked list to support your explanation.

1. Find the node before the target (prevTarget)
   Traverse the list until prevTarget->next == target.
2. Detach the target node
   Skip it by re-linking:

prevTarget->next = target->next;

3. Insert the target node after destination
   Link target into its new position:
   target->next = destination->next;
   destination->next = target;
4. Edge cases:
   If the target node is the head → use a dummy node before the head.
   If the target is next to the destination → handle pointer adjustment carefully.

③ Is there any **variation of a linked list** (e.g., doubly linked list or circular linked list) that can simplify or improve this operation?

A **doubly linked list** or **circular linked list** can make the MoveTo operation more efficient since each node has **direct access to its predecessor**.

Q2: (40 pts, 10 pts for each) **Definition of Data Structures**
Define the following data structures and list their fundamental operations.
A2:

① Definition of "Stack"

A stack is a linear data structure that follows the **Last In, First Out (LIFO)** principle. The last element added is the first to be removed.

② Definition of "Queue"

A queue is a linear data structure that follows the **First In, First Out (FIFO)** principle. The first element added is the first to be removed.

③ Preliminary operations of "Stack"

- push(): Insert an element onto the top of the stack.
- pop(): Remove the top element.
- peek() / top(): View the top element without removing it.
- isEmpty(): Check if the stack is empty.
- isFull(): Check if the stack is full.

④ Preliminary operations of "Queues"

- enqueue()/delete(): Add an element to the rear.
- dequeue()/addQ(): Remove an element from the front.
- front(): Access the front element without removal.
- isEmpty(): Check if the queue is empty.
- isFull(): Check if the queue is full.

Q3: (30 pts) **AI Copilot Application**

Choose **up to two** data structures from the visualization list above.

Compose a **single prompt (within 300 words)** that you would use with an **AI Copilot** to explore or learn advanced concepts related to your chosen data structures.

A3:

I'm a sophomore majoring in computer science, currently taking a course in **Data Structures**. I have a basic understanding of **C++ programming**. I'd like help learning how to apply **arrays** and **linked lists** in **real-world applications**, starting from **easy** to **medium-level** examples. Please guide me through practical use cases, implementation strategies, and how these data structures solve problems in real scenarios.

Prompt template

I'm a [year level] student majoring in [major], currently taking a course in **Data Structures**. I have a basic understanding of [programming language]. I'd like help learning how to apply the data structure of **[specific topic, e.g., arrays or linked lists]** in **real-world applications**, starting from **easy** to **medium-level** examples. Please guide me through:

- Practical use cases
- Implementation strategies
- How this data structure solves problems in real scenarios