



<script src="./themes/hover-line.js"></script>

# Tutorial 3: Linear Data Structures and Search Algorithms

**CAB301 - Algorithms and Complexity**

School of Computer Science, Faculty of Science

# Agenda

## 1. Recap:

- i. Linear Data Structures
- ii. Search Algorithms

## 2. Tutorial Questions:

- i. **Part A:** Linear Data Structure - Stack
- ii. **Part B:** Searching Algorithms - Binary Search
- iii. **Part C:** Programming tasks - Stack, Circular Linked List, Collections of Custom Objects

# Recap: Linear Data Structures

A **data structure** stores and organises data so that it can be accessed and modified efficiently.

Provides operations such as **insertion**, **deletion**, **searching**, and **sorting**.

## Linear Data Structure:

- Elements stored in a sequence
- Except for first and last, each element has a unique predecessor and successor
- *Examples:* Array, Linked List, Stack, Queue

```
int[] numbers = new int[5] { 1, 2, 3, 4, 5 };
```

# Recap: Search Algorithms

**Searching** is used to find an element in a collection of elements, to either:

- Confirm if the element exists
- Get the key (e.g., index) of the element

	Sequential Search	Binary Search
<b>Idea</b>	Go through each element one by one from start	Reduce the search space by half each time
<b>Time Complexity</b>	$\mathcal{O}(n)$	$\mathcal{O}(\log n)$

# Part A - Question 1: Stack

Perform the following operations on a **Stack** with a capacity of 6:

<div style="display: flex"> <div style="flex: 0.5" class="hover-line">

**Enqueue(4)**

**Enqueue(1)**

**Enqueue(3)**

**Dequeue()**

**Enqueue(8)**

**Dequeue()**

## Part B - Question 2: Binary Search

<small style="margin: 0 -60px; font-size: 22px; display: flex;" > <div style="flex: 1" class="hover-line">

**ALGORITHM** BinarySearch( $A[0..n - 1]$ ,  $K$ )

$l \leftarrow 0; r \leftarrow n - 1$

**while**  $l \leq r$  **do**

$m \leftarrow \lfloor (l + r) / 2 \rfloor$

**if**  $A[m] = K$  **then**

**return**  $m$

**else if**  $A[m] < K$  **then**

TEQSA Provider ID PRV16013, Queensland University of Technology, CRICOS No. 00099G

# Part C - Question 3: Stack Implementation

Implement a **Stack** in C# using an **Array**.

Provides the following:

- **Size** : number of elements in the stack
- **Empty** : **true** if the stack is empty, **false** otherwise
- **Push** : add an element to the top of the stack
- **Pop** : remove and return the top element of the stack
- **Peek** : return the top element of the stack without removing it

Use the skeleton provided.

# Part C - Question 4: Circular Linked List

Implement a **Circular Linked List** in C#, from the following interface:

```
public interface IQueue {  
    int Capacity { get; }  
    int Count { get; }  
    bool IsEmpty();  
    bool IsFull();  
    void Enqueue(int value);  
    Object Dequeue();  
    Object Head();  
    void Clear();  
}
```



# Part C - Question 5: Custom Objects

Create a `CustomerCollection` class in C# that stores `Customer` objects (with `FirstName`, `LastName`, and `Phone`).

Implement the following operations:

- `Find(string firstName, string lastName)`: returns the associated `Phone` number
- `Insert(string firstName, string lastName)`
- `Insert(Customer customer)`
- `Delete(string firstName, string lastName)`
- `Display()`: prints all the `Customer` objects