# Homework 2

CS201: Data Structures and Algorithms, Fall 2021
**Due: 22/11/2021, 11:59pm**

**Topic:** Basic Data Structures, Recurrance Relations, Proof by induction.

## Problem 1

Implement the following basic data structures. For each API (or function), discuss its time complexity .

### Stack (5 Points):

Implement a `1.1 Stack` data structure that can be accessed using the following APIs

- `isEmpty:` takes no parameters and return `TRUE` if the stack is empty.
- `push:` takes an integer as a parameter and push it on the stack.
- `pop:` takes no parameter, remove top integer $x$ from the stack, and return $x$. If stack is empty, raise an exception
- `top:` takes no parameter, return top integer $x$ from the stack. If stack is empty, raise an exception
- `size:` takes no parameter, and return the number of integers/elements currently on the stack.

### 1.2 Queue (5 Points):

Implement a `Queue` data structure that can be accessed using the following APIs

- `isEmpty:` takes no parameters and return `TRUE` if queue is empty.
- `enqueue:` takes an integer as a parameter and append it to the tail of the queue.
- `dequeue:` takes no parameter, remove front integer $x$ from the queue, and return $x$. If queue is empty, raise an exception.
- `top:` takes no parameter, return front integer $x$ from the queue. If queue is empty, print "ERROR: queue is EMPTY".
- `size:` takes no parameter, and return the number of integers/elements currently on the queue.

### 1.3 LinkedList (5 Points):

Implement a `LinkedList` data structure that can be accessed using the following APIs

- `isEmpty:` takes no parameters and return `TRUE` if linkedlist is empty.
- `add:` takes an integer as a parameter and append it to the tail of the linkedlist.
- `remove:` takes an integer $x$ as a parameter and remove it from the linkedlist. If $x$ not found, raise an exception
- `search:` takes an integer $x$ as a parameter and return `TRUE` if $x$ is on the list.
- `size:` takes no parameter, and return the number of integers/elements currently on the linkedlist.

## Problem 2 (15 Points):

Implement a Queue using 4 stacks. State the time-complexity for each queue API (i.e. `isEmpty, enqueue, dequeue, size, top`.)

## Problem 3 (15 Points):

Implement a Stack using 2 queues. State the time-complexity for each stack API (i.e. `isEmpty, push, pop, size, top`.)

## Problem 4 (25 points):

Write the function `Merge(LinkedList L1, LinkedList L2)` that takes two sorted LinkedLists `L1, L2` and merge them into one. Use only nodes in `L1, L2`, you are **NOT** allowed to create new nodes or use temporary arrays. State the time complexity of your function.

### Example 1

**input:** `L1 = 1->2->3, L2= 1->3->4->6`
**output:**
`1->1->2->3->3->4->6`.

### Example 2

**input:** `L1 = 1, L2= 1`
**output:**
`1->1`.

## Problem 5 (30 Points):

A balanced parentheses is a string defined as follows:

$S = (S')$ where $S'$ is either an empty string or a balanced parentheses,
$S = A + B$ where `A` and `B` are balanced parentheses and `A+B` is their concatenation.

Let $S = P_1 + P_2 + \ldots + P_n$ be a balanced parentheses or a concatenation of balanced parentheses where $P_i$ is a balanced parentheses and $n \geq 1$. Implement the procedure `split(String S)` that prints each balanced parentheses $P_i$ on a separate line. State the time complexity of your implementations.

### Example 1

**input:** `S = ()(()())()`
**output:**
`()`
`(()())`
`()`
**Explanation:**
Each of `()`, `(()())`, `()` is a balanced parenthesis.

**Example 2**

**input:** `S = (()()()()())`
**output:**
`(()()()()())`
**Explanation:**
`S` contains only one balanced parantheses `(()()()()())` (not a concatenation of balanced parantheses.)

*Hint: Representing the input data in a specific structure can make the problem easier to solve.*

**Deliverables:**

- Submit **one source file** that includes your solution for the **two problems.**
- Add comments that explains your code.
- Calculate **the time complexity** of your solution and add it as a comment in the source file.
- You may program in `C++, Java,` or `Python`
- **NOTE:** You might be selected to present your solution to the TA.

**Hints and Advice:**

- Start working on the assignment ASAP.
- Anyone who passed CS142 shall be able to solve this assignment.
- Ask questions if you don't know.