Cairo University
Faculty of Engineering
Computer Engineering
Department

# Data Structures and Algorithms
# Lab Exam

## Question 1 (Stacks and Queues)

Implement the following **global** function in the file "Problem1.cpp" without changing the function header:

**void ExampleFunc (LinkedQueue<int> & input)**

- Read all test cases given before implementation, and consider any corner cases in your implementation.

## [ Implementation Constraints ] :

**- Note:** No assignment operator is defined, so don't use for example: queue1 = queue2; but instead loop and dequeue from queue2 and enqueue to queue1. Same for Stacks.

- You are allowed to use auxiliary (temp) stacks(s)/queue(s) as many as you want.

- You are NOT allowed to use any other array-like or linked-list-like data structures.

- You are allowed to write code ONLY inside the { } of the required global function. Don't change anything else in the code, even the prototypes.

- You are NOT allowed to add any helper functions or call any existing functions other than the global function required of you.

- Don't print anything inside the function.

- If the function returned a value, make sure that all function paths return a value (i.e., no combination of if or else conditions does not returns a value). ➔
[VERY IMPORTANAT] This will make a syntax error in the online judge and no run.

**A manual penalty will be applied to your grade if you violate the above implementation constraints**.

## Question 2: Trees

Implement the following **private recursive** function in the file “Problem2.cpp” without changing the function prototype:

**int recExampleFunc ( TNode*& subRoot )**

- Read all test cases given below before implementation, and consider any corner cases in your implementation.

## [ Implementation Constraints ] :

- You are allowed to write code ONLY inside the { } of the required private function **and the { } of its corresponding public function (if needed).** Don’t change anything else in the code, even the prototypes.

- You are NOT allowed to use any auxiliary array-like or linked-list-like storage data structures, not even stacks, queues, or “strings”.

- You are NOT allowed to add any helper functions or call any existing functions other than the recursive function required from you.

- You MUST make the requirement using a single traversal to the tree.

- Use pre-order traversal (i.e., the recursive calls of left and right are at the end after handling the subRoot first).

- Don’t print anything inside the function.

- If the function returned a value, make sure that all function paths return a value (i.e., no combination of if or else conditions does not return a value). ➔
[VERY IMPORTANAT] This will make a syntax error in the online judge and no run.

**A manual penalty will be applied to your grade if you violate the above implementation constraints**.