# COMP3506/COMP7505—Algorithms and Data Structures

School of Information Technology and Electrical Engineering

# Week 5 Tutorial

**Question 1**

Give a big-O characterization, in terms of $n$, of the running time of the following algorithm.

**Algorithm** SumEven

**Input**: An array $A$ storing $n >= 1$ integers.

**Output**: The sum of the elements at even cells in $A$.

$s \leftarrow A[0]$

for $i \leftarrow 2$ to $n - 1$ by increments of 2 do

    $s \leftarrow s + A[i]$

**return** $s$

**Question 2**

Algorithm A executes an $O(\log n)$-time computation for each entry of an $n$-element array. What is the worst case running time of Algorithm A?

**Question 3**

Given an $n$-element array $X$, Algorithm B chooses $\log n$ elements in $X$ at random and executes an $O(n)$-time calculation for each. What is the worst-case running time of Algorithm B?

**Question 4**

Suppose you have a deque $D$ containing the numbers (1, 2, 3, 4, 5, 6, 7, 8), in this order. Suppose further that you have an initially empty queue $Q$. Give a pseudo-code description of a method that uses only $D$ and $Q$ (and no other variables or objects) and results in $D$ storing the elements (1,2,3,5,4,6,7,8), in this order.

**Question 5**

Describe how to implement the stack ADT using two queues. What is the running time of the push() and pop() methods in this case?

**Question 6**

Draw an arithmetic expression tree that has four external nodes, storing the numbers 1, 5, 6, and 7, (with each number stored in a distinct external node, but not necessarily in this order), and has three internal nodes, each storing an operator from the set {+, −, ×, /}, so that each value of the root is 21. The operators may return and act on fractions, and an operator may be used more than once.

**Question 7**

Draw a (single) binary tree *T* such that

- Each internal node of *T* stores a single character
- A *preorder* traversal of *T* yields EXAMFUN
- An *inorder* traversal of *T* yields MAFXUEN

**Question 8**

Describe an algorithm for computing the number of descendents of each node of a binary tree, based on the Euler tour traversal.

How would you override the abstract visit methods of the class presented in lectures?