University College Dublin
An Coláiste Ollscoile, Baile Átha Cliath

SEMESTER I EXAMINATIONS
ACADEMIC YEAR 2017/2018
SAMPLE EXAM PAPER

COMP 47500

Advanced Data Structures in Java

Prof. J. Pitt

Prof. P. Cunningham

Assoc. Prof. Eleni Mangina*

Time allowed: 2 hours

Instructions for candidates

Answer **one question from part A and one question from part B.**

All questions carry equal marks. The paper is marked out of 100.

.

Part A:

1. Answer parts (a) to (c). 50 marks in total

(a) (15 marks)

(i) What is an ArrayList? List the methods commonly associated with the ArrayList. **(5 marks)**

(ii) Briefly define the following terms: Wrapper class, autoboxing and enhanced "for" loop and provide examples for each one. **(10 marks)**

(b) (15 marks)

(i) What is a *Stack*? List the operations commonly associated with the Stack Abstract Data Type (ADT). **(5 marks)**

(ii) Describe how a Stack can be used to evaluate Arithmetic Expressions like:
14 - 3 * 2 + 7 = (14 - (3 * 2)) + 7
(5 marks)

(iii) Describe how to implement two stacks using one array. The total number of elements in both stacks is limited by the array length; all stack operations should run in O(1) time. **(5 marks)**

(c) (20 marks)

(i) What is a *Queue*? List, giving a brief description for each one, the operations commonly associated with the *Queue* Abstract Data Type (ADT). **(5 marks).**

(ii) What is a *Deque*? Give the pseudo-code algorithm for insertion into the front of a *Deque*. **(5 marks)**

(iii) Describe in pseudo code a linear-time algorithm for reversing a *Queue Q*. To access the *Queue*, you are only allowed to use the methods of *Queue* ADT. *Hint: Consider using a Stack.*
(5 marks)

(iv) Another variant of the basic Queue Abstract Data Type is a *Priority Queue*. Explain how *Priority Queue* works. **(5 marks)**

2. Answer parts (a) to (c). 50 marks in total

(a) Explain how you would perform an experimental evaluation of the array-based implementations of the insertion sort and selection sort algorithms. Give some outline code that illustrates your answer – you do not need to write the actual sort algorithms or the data set creation code, instead, you can use comments of the form "`// insertion sort of array X`". **(15 marks)**

(b) An efficient algorithm can significantly reduce the time taken to complete long running computations. Compared to Experimental Analysis, briefly describe the Asymptotic Algorithm Analysis.

(5 marks)

(c) **(30 marks)**

(i) *Merge Sort* and *Quick Sort* are examples of sorting algorithms that are based upon an algorithmic design pattern called divide-and-conquer. Describe this design pattern, and use it to outline the main steps involved in both the *Merge Sort* and *Quick Sort* algorithms. **(10 marks)**

(ii) The key operation of the *Merge Sort* algorithm is the *merge operation*. Give the pseudo code for both this operation and the overall merge sort algorithm. **(10 Marks)**

(iii) The key operation of the *Quick Sort* algorithm is the *partition operation*. Illustrate, using a quick sort tree, how the quick sort algorithm would sort the following items: **(10 Marks)**

31, 21, 1, 4, 8, 22, 13, 44, 10, 28

Part B:

1. Answer parts (a) to (f). (50 marks in total)

(a) What is a *proper Binary Tree*? List and describe any operations are commonly associated with this type of tree. These operations should not be associated with the *Binary Tree* ADT. **(8 marks)**

(b) What is a *total order relation*? How are total order relations used to implement *binary search trees*? **(8 marks).**

(c) *Proper Binary Trees* can be used to implement *binary search trees*. Explain how they are used and why they are a good implementation strategy. **(5 marks).**

(d) Give the pseudo code for the *find* operation of a binary search tree. **(8 marks).**

(e) Explain how a value is removed from a *binary search tree*. Illustrate your answer with examples to cover the three cases that may rise when deleting a node *v* from the tree. **(15 marks).**

(f) Explain how you could use a *Binary Search Tree* to implement a *map*. Explain how the performance of a *tree-based map* would compare to a *hash map*. What benefit does a *tree based map* have over a *hash map*? **(6 marks).**

2. Answer parts (a) to (c). (50 marks in total)

(a) Given a graph with *n* vertices and *m* edges, what is the upper bound for *m* in an undirected graph with no self loops and multiple edges. **(5 marks).**

(b) There are three main approaches to implementing a *graph*. Describe each approach, illustrating your answer with a diagram that highlights the design underlying that approach. **(15 marks).**

(c) Compare the performance of each of the approaches outlined in (b) with respect to the following issues:

i. Space (Memory)
ii. incidentEdges(v) method
iii. areAdjacent(v,w) method
iv. insertVertex(o) method
v. insertEdge(v, w, o) method
vi. removeVertex(v) method
vii. removeEdge(e) method

(5 marks).

(d) What is a Minimal Spanning Tree? Describe and prove the cycle and partition properties of Minimal Spanning Trees.

(10 marks).

(e) Give pseudo code for Kruskal's Algorithm for finding the *Minimal Spanning Tree* of a weighted graph, and use this algorithm to find the *Minimal Spanning Tree* of the graph below.

(15 marks).