

**Faculty of Science**

**Open-Book and Remote Assessment Cover Page**

**Student Name: Rachel Doogue**

**Student Number: C00237335**

**Lecturer Name: Áine Byrne**

**Course: BSc in Software Development**

**Module: DSA II (Data Structures and Algorithms)**

**Stage/Year: 3**

**Date: March 2nd 2021 9.30am**

**Time: 1 hour allowed**

**Instructions: Answer ALL Questions**

|  |  |  |
| --- | --- | --- |
| **Marks** | **Out of 20** | **Comments if required** |
| Q1 |  |  |
| Q2 |  |  |
| Q3 |  |  |
| Total | % |  |

**Assessment Details:**

* This is a remote, online assessment- you may use your notes/previous examples.
* You are to complete the tasks individually and independently; no online or external assistance is permitted.
  + When you submit, you will be accepting a declaration of academic integrity.
* You may be required to orally defend any of your answers, to the lecturer, at a given time after the assessment has been completed, as outlined in the student regulations.

**To Submit:**

* A OneDrive link for uploading your exam submission.

<https://instituteoftechnol663-my.sharepoint.com/:f:/g/personal/byrnea_itcarlow_ie/Eh0N-bKw_XpIjM85nqYKQ2wBsT-bqDzNJP7eDfsboJWHCg>

* You may also email me your exam submission [aine.byrne@itcarlow.ie](mailto:aine.byrne@itcarlow.ie)

Many of the questions require you to use the following three numbers.

Using your student number. Take the last 6 digits and split them into three numbers as follows.

For example: Student number C00123456 will split into numbers 12, 34, 56.

**Q1. (3 parts)**

(a) Add the three numbers from your student number to the following list of numbers to be sorted (if duplicates, remove one). Execute the Quicksort algorithm up to the first partition stage.

45, 74, 8, 58, 3, 21, 73, 89

Indicate clearly which item is been chosen as the pivot and show all swaps required.  (6)

(b) Given a linear singly linked list called **L** and a head pointer **Head** which points to the first element in L. Write the algorithms for the following operations: (8)

**DelFirst** (deletes the first node)

**DelNext** (given pointer p, deletes the node after the node pointed to by p)

**DelPrevious** (given pointer p, deletes the node before the node pointed to by p)

**FindLast** (deletes the last node)

Write the Big (O) value of each of the algorithms.

(c) Given the following algorithm **FunS():** (6)

**int FunS(int n)**

**{**

**Output n ;**

**if (n > 0)**

**return FunS(n-2) ;**

**}**

1. Using the last digit of your student number- if its less than 5 add 4 to it.

Trace this algorithm with the last digit of your student number. What’s output?

(ii) Write an iterative version of this method.

**Q2.**

(a) Draw a **Max-Heap** with exactly ten nodes. Put a different value in each node. Using the same ten values, draw a min-heap.

Show how both heaps are represented in an array. (5)

(b) Add another number into your **Max-Heap**. Choose a number which will involve at least one swap.

Draw the resulting heap. (2)

(c) Given a **Max-Heap** of integers **H** and an integer **n** which is the number of nodes in the heap, write the **algorithms** for the following heap operations,

**Parent(p)**: returns the node value of the parent of a node at index p

**NumParents()**: returns the number of parents (non-leaf nodes) in the heap H

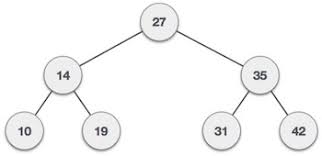
**DisplayAllLeaves()**: Displays the node values of all leaves in the heap H

(9)

(d) Perform the first 2 passes of the HeapSort on your heap from part(b). Draw the resulting heap after each pass. (4)

**Q3.**

(a) Consider the Binary Search Tree (BST) below: (5)



1. Add the three numbers from your student number into the BST above
2. Using the updated tree, Suppose we remove the root, replacing it with an element from the right tree. What will be the new root? Draw the resulting tree.

(b) Describe one representation as to how a General Tree (unlimited number of children) could be implemented. Use diagrams to illustrate your answer.  (2)

(c)  Using your implementation method from part (c); Write the algorithms for the following operations:

**Is\_Parent(p):** returns whether a given node p is a parent in the tree.

**Level1(root):** outputs the value of the children of the root node

**BigChild(p):** returns the value of the largest child of a given node p

(8)

(d) Assume the integers [2, 5, 12, 8, 10, 13, 14, 4] are added to an AVL tree. Sometimes, adding an item causes a rotation to take place. Every time a rotation takes place, note the item and the point where the imbalance occurs. Submit your solution as a list of tuples, one for each rotation, containing the item added and the imbalanced node. (5)