Faculty of Engineering, Environment and Computing

210CT Programming, Algorithms and Data Structures

Assignment Brief 2018/19

Module Title Programming, Algorithms and Data Structures	Ind/Group Individual	Cohort (Sept/Jan/May) September	Module Code 210CT
Coursework Title (e.g. CWK1) Individual Programming Assignment			Hand out date: 22.10.2018
Lecturer Dr. Diana Hintea			Due date: 30.11.2018
Estimated Time (hrs): 60 Word Limit*: Not applicable	Coursework type: Individual Programming Assignment		% of Module Mark 30%

Submission arrangement online via CUMoodle: 6pm submission on the 30th of November 2018
File types and method of recording: Word or PDF
Mark and Feedback date: 2 weeks after VIVA
Mark and Feedback method: VIVA with a marking scheme and feedback sheet

Module Learning Outcomes Assessed:

On successful completion of this module a student should be able to:

1. Understand and select appropriate algorithms for solving a range of problems.

2. Design and implement algorithms and data structures for novel problems.

3. Reason about the complexity and efficiency of algorithms.

4. Demonstrate the use of object oriented programming features and advanced language features such as events, unit testing and GUIs.

Task and Mark distribution

1. Build a Binary Search Tree (BST) to hold English language words in its nodes. Use a paragraph of any text in order to extract words and to determine their frequencies.
   Input: You should read the paragraph from a suitable file format, such as .txt. The following tree operations should be implemented: a) Printing the tree in pre-order. b) Finding a word. Regardless whether found or not found your program should output the path traversed in determining the answer, followed by yes if found or no, respectively.

2. Implement a function that deletes a node in a binary search tree in a language of your choice.

3. Implement the structure for an unweighted, undirected graph G, where nodes consist of positive integers. Also, implement a function isPath(v, w), where v and w are vertices in the graph, to check if there is a path between the two nodes. The path found will be printed to a text file as a sequence of integer numbers (the node values).

4. Using the graph structure previously implemented, implement a function isConnected(G) to check whether or not the graph is strongly connected. The output should be simply 'yes' or 'no'.

5. Implement BFS and DFS traversals for the above graph structure. Save the nodes traversed in sequence to a text file.

6. Adapt the previous graph structure to support weighted connections and implement Dijkstra’s algorithm.

Notes:

1. You are expected to use the CUHarvard referencing format. For support and advice on how this students can contact Centre for Academic Writing (CAW).
2. Please notify your registry course support team and module leader for disability support.
3. Any student requiring an extension or deferral should follow the university process as outlined here.
4. The University cannot take responsibility for any coursework lost or corrupted on disks, laptops or personal computer. Students should therefore regularly back-up any work and are advised to save it on the University system.
5. If there are technical or performance issues that prevent students submitting coursework through the online coursework submission system on the day of a coursework deadline, an appropriate extension to the coursework submission deadline will be agreed. This extension will normally be 24 hours or the next working day if the deadline falls on a Friday or over the weekend period. This will be communicated via email and as a CUMoodle announcement.
6. Please only submit text-based code for the official online submission (no screenshots) otherwise a mark of 0 will be awarded.