2016/2017

# Programming, Algorithms and Data Structures (210CT)

Coursework

**MODULE LEADER:** Dr. Diana Hintea

**Student Name:**

**SID:**

**I can confirm that all work submitted is my own: Yes** ☐

This coursework is testing your ability to design algorithms, write pseudocode, describe their efficiency, implement various data structures and use the programming language of your choice for the implementation.

## Coursework

### Basic

#### Week 1

1. Write a function that randomly shuffles an array of integers and explain the rationale behind its implementation.

2. Count the number of trailing 0s in a factorial number.

#### Week 2

3. Write the pseudocode for a function which returns the highest perfect square which is less or equal to its parameter (a positive integer). Implement this in a programming language of your choice.

4. Look back at last week's tasks. Describe the run-time bounds of these algorithms using Big O notation.

5. Write the pseudocode corresponding to functions for addition, subtraction and multiplication of two matrices, and then compute A=B*C –2*(B+C), where B and C are two quadratic matrices of order n. What is the run-time?

#### Week 3

6. Write the pseudocode and code for a function that reverses the words in a sentence. Input: "This is awesome" Output: "awesome is This". Give the Big O notation.

7. Write a recursive function (pseudocode and code) to check if a number n is prime (hint: check whether n is divisible by any number below n).

8. Write a recursive function (pseudocode and code) that removes all vocals from a given string s. Input: "beautiful" Output: "btfl".

#### Week 4

9. Adapt the binary search algorithm so that instead of outputting whether a specific value was found, it outputs whether a value within a specific interval was found.

#### Week 5

10. Given a sequence of n integer numbers, extract the sub-sequence of maximum length which is in ascending order.

11. Based on the Python code or the C++ code provided in class as a starting point, implement the binary search tree node delete function.

#### Week 6

12. Implement TREE_SORT algorithm in a language of your choice, but make sure that the INORDER function is implemented iteratively.

### Week 7

13. Write the pseudocode for an unweighted graph data structure. You either use an adjacency matrix or an adjacency list approach. Also, write a function to add a new node and a function to add an edge. Following that, implement the graph you have designed in the programming language of your choice. You may use your own linked list from previous labs, the STL LL, or use a simple fixed size array (10 elements would be fine).

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

### Week 8

15. Implement Dijkstra's algorithm for a weighted graph data structure (you have to update your previous data structure so that it can deal with weights).

## Advanced

1. Write a program to predict the number of creatures in a fictional alien invasion. An alien lays X eggs each day (there are no genders in this species) and the eggs hatch after Y days. If X is 3 and Y is 5, how many aliens will there be 30 days after a single alien invades?

2. A sparse matrix is a matrix where the number of elements which are zero is bigger than the number of elements which are not zero. Find a way to store sparse matrices, and write the functions to add, subtract, and multiply pairs of such matrices. Do not use predefined functions for the operations on matrices in your programming language of choice.

3. Given two strings of n and m integer elements, write the pseudocode to compute:
   a) The string that contains all the elements belonging to both strings.
   b) The string of all the elements of the two given strings, written once.
   c) The string of the elements from the first string, without the elements that are also in the second string.
   What's the run time?

4. Write the pseudocode for a recursive program to generate the Cartesian product (product set, direct product, cross product) of n sets.

5. In addition to the normal homework task, write a function that takes four parameters representing the constant and multiplier of two linearly growing (as in $O(m \times n + k)$ ) functions and determines the critical value of n (which should be an integer) at which the relative run-time of the two algorithms switches. That is, at which input size is algorithm A slower than B and at which is B slower than A? Use an iterative approach rather than solving the equations.

6. Consider having n cubes, each being characterized by their edge length and their colour. Use the cubes (not necessarily all of them) in order to build a tower of maximum height, under the following conditions:
   a) any two neighbouring cubes must be of different colours.
   b) the edge length of a cube is lower than the edge length of the cube placed below it.

7. Let's consider a labyrinth as a n × m matrix, where the corridors are denoted by 1s situated in consecutive positions on the same line or column. The rest of the elements are 0. Within the labyrinth, a person is considered to be in position (i, j). Write a program that lists all exit routes which do not pass the same place twice. Input: n, m, the rows of the matrix, the coordinates of the exit and the coordinates of the person (row, column). Output: a sequence of row/column pairs representing the person's successive position.

8. Adapt the quick sort algorithm to find the mth smallest element out of a sequence of n integers.

9. Write a function to calculate the kth power of a square matrix, using pointers to access to the elements of the matrix. The resulted matrix will be displayed in natural form.

10. Using the model of a circular single-linked list, implement the following scenario: N children stand in a circle; one of the children starts counting the others clockwise. Every Nth child leaves the game. The winner is the one who remains. Notes: Read the number of children, the childrens' names and the one starting to count from the standard input. Input: 4; Diana, Michael, David, Mary; Start: Diana; Winner: Michael.

11. 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 words and frequencies from a file in a suitable format, such as .csv. The following tree operations should be implemented: a) Listing (word, frequency) pairs for each of the tree nodes. b) Printing the tree in preorder. C) 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.

12. 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).

13. 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'.

14. Consider the structure of a directed weighed graph. Implement an algorithm to find its maximum cost spanning tree. The output should be the preorder and inorder traversal of the tree. Describe the running time of this algorithm.

## Submission Date

**Cut-off date:** You should submit your completed assignment to arrive no later than the cut-off date: **2/12/2016.** Write your solutions to this coursework in a single word-processed document (you could also submit as a PDF). Do not include any screenshots of your code (it has to be text) in your .doc or .pdf otherwise it won't be considered for marking. Each page should be numbered and headed with the title of the coursework, your name and your student id number, for example:

210CT Coursework Your Name 2122222

Additional instructions on submission can be found in Appendix 1.

## Plagiarism Note

As with all assessed work, both the research and written submission should be your own work. When submitting this work you are explicitly indicating that you have read the rules on plagiarism as defined in the University regulations and that all work is, in fact, your own, except were explicitly referenced using the accepted referencing style.

## Objectives

This particular coursework covers learning outcomes 1, 2, 3 and 4 as shown below:

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.

## Appendix 1: Submission Instructions and Extensions

**Follow these instructions carefully. Failure to do so will mean sections or all of the coursework marked being zeroed.**

1. The coursework should be submitted as an online submission. Your submission should be either as a MS Word or PDF document.

2. Your submission should not contain any screenshots taken of your code, but the code itself.

3. Your submission will consist of the code written to solve the given tasks, together with a brief explanation on how this was achieved. Overall, the report should be well documented and referenced.

4. You should attempt either the basic or the advanced homework for each week, therefore you can combine the two throughout the coursework. You will not get extra marks if you attempt the advance coursework!

5. You have to include the GitHub link to your repository in the coursework submission.

6. Your assessment will be a VIVA.

7. Extensions will only be given if supporting evidence is produced, so only ask for an extension if you really need it and if you have no supporting evidence to support your request it will be rejected. Any such request should be directed to Registry.