Computer Science and Creative Technologies

Coursework or Assessment Specification

Module Details

Module Code	UFCFW4-30-2
Module Title	Design and Analysis of Data Structures & Algorithms
Module Leader	Dr Gordon Downie & Dr Elias Pimenidis
Module Tutors	Dr Kun Wei, Dr James Lear, and Mr. Jacob Baker
Year	2020-21
Component/Element Number	B / 2
Total number of assessments for this module	2
Weighting	60%
Element Description	Coursework B

Dates

Date issued to students	22 February 2021
Date to be returned to students	24 May 2021
Submission Date	22 April 2021
Submission Place	Blackboard - Online
Submission Time	14:00
Submission Notes	Please submit all files in a zipped folder **clearly labelled "DADSA2021B – Your Name"**

Feedback

Feedback provision will be	Formative / Verbal, during practical sessions. Written on return of the marked work.

Contents

Section 1: Overview of Assessment

This assignment assesses the following module learning outcomes:
Analyse requirements and select appropriate solutions
Design programmes that use appropriate data structures.
Implement data structures and the algorithms that maintain them, allowing for secure processing of the data.

The assignment is worth ***30%*** of the overall mark for the module.
Broadly speaking, the assignment requires you to select and implement appropriate data structures, design and implement algorithms and create the relevant software applications that will allow a user to store, update and manipulate the data relating to the operations of an animal sanctuary.
The assignment is described in more detail in section 2.
This is an individual assignment.
Working on this assignment will help you to develop the ability to relate organisational requirements into design choices for the storage and management of data. It will also help you practice your programming skills on Python and explore means by which the efficiency of your programming can improve.
If you have questions about this assignment, please ask your practical session tutor for clarifications.
Python libraries should not be used in creating code, neither should built-in data structure and algorithms templates should be utilised. The only readily available code you can reuse and customise is that for importing and reading a CSV file.
You can build your own data structures using the Python default structures as building blocks – see example of building a linked list using the default Python array (week 3 TB1 practical worksheet) and examples of building trees and graphs can be found in the worksheets for practical sessions for weeks 2-5 of TB2.

Section 2: Task Specification

Referring CCU Patients to Dietitians

Critical Care Units in NHS hospitals are some of the busiest wards at any time of the year. Patients that arrive there are usually in serious need of medical care. Beyond the administration of any medication and the carrying out of any medical procedures, staff at the CCUs need to ensure patients are fed properly and according to their condition. Therefore it is considered of high importance that patients are assessed by dietitians upon arrival at a CCU. As with many cases in hospitals dietitians are a rare resources in the Great Western National Hospital (GWNH). To ensure that the dietitians get to see those patients that are mostly in need of a proper evaluation, nurses at the CCU apply a rule of thumb initial assessment to prioritize those mostly in need. This is often prone to error though and may put some patients at risk, if they are not properly assessed by a dietitian when initially admitted at the CCU.
The hospital has asked our help to automate the selection and create a ranking in terms of the order of priority that patients should be referred to a dietitian.

The rule for assessing a referral is based on the following condition table.

Condition	**Decision**
Obese OR Underweight	Refer
Hypertension	Refer
Asthmatic OR Smoker	Refer
NJT OR NGR	Refer
Renal Replacement Therapy	Refer
Ileostomy / Colostomy	Refer
Parenteral Nutrition	Refer

If a patient is asthmatic OR a smoker and is over 55, **OR** Obese & suffers from hypertension, they should be ranked as top priority for referral.
Also top priority for referral should be those that have any other combination of more than two conditions. Where two patients have exactly the same level of priority the order in which they need to be seen by the dietitians will be determined by their age. The older the person the higher the urgency to be assessed.

A patient's condition as underweight, normal, overweight, or obese is based on their BMI (Body Mass Index) and is determined according to the following table.

Build	**BMI**	**Condition**	**BMI**	**Condition**	**BMI**	**Condition**	**BMI**	**Condition**
Slim	< 18.5	Underweight	18.5-25	Normal	25-28	Overweight	28+	Obese
Regular	< 18.5	Underweight	18.5-25	Normal	25-29	Overweight	29+	Obese
Athletic	< 18.5	Underweight	18.5-25	Normal	25-30	Overweight	30+	Obese

A patient's BMI is calculated using the following formula

$$\textbf{BMI = weight (Kgs) / Height}^2 \textbf{ (m)}$$

Task 1 – Design (pseudocode & diagrams) and implement software in Python to deliver the following results.

Use the table patient data to

Task 1

- Import the data into a data structure that you will create
- Calculate the BMI for each patient and classify these patients as underweight, normal, overweight, or obese.
- Update the data structure adding the BMI
- Display / Print on screen the patient name, age, BMI, and weight classification. Sort the display so that those classed as obese are shown at the top of the screen , followed by those classed as underweight, then those that are overweight with the list completed by those that are classed as normal body mass. Insert breaks every 10 patients to allow the user of the system to study the results.
- Display / Print on screen the worst 5 underweight and the worst 5 obese patients in two groupings, male and female.

Task 2

- Establish which patients need to be referred to a dietitian.
- Rank the order of priority according to the rules given above.
- Display / Print on Screen patient names of those that need to be referred to a dietitian. Insert breaks every 10 patients to allow the user of the system to study the results.

Task 3

- Justify your design choices (data structures & algorithms) in terms of effectiveness and efficiency of the system that you have produced.
- Discuss any issues relating to data protection and GDPR that the system's users should be concerned about.
- For this task, produce one document of 400 words maximum.

Section 3: Deliverables

One folder in zip format (only) must be uploaded via the relevant link on the module's space on Blackboard. The link will be available two weeks before the due date and will be communicated to students via an email announcement.
You can reuse code and design that you have produced for one of the tasks here to fulfil the requirements of another task.
The folder must contain:

- The software modules in Python clearly labelled.
- A word / text file that provides clear instructions as to how to run the program – This is particularly important where more than one Python files have been submitted.
- All the data files packaged in a way that the software will run and access them without any problems.
- A short video demonstrating your software running and fulfilling each of the tasks in the correct order.
- A word / PDF file for task 3.
- A word /PDF file with your pseudocode
- Design diagrams – these should be accessible through software available at UWE.

All program files saved in Python format (3.6 - 3.8) – any other version will not be accepted and the work will not be marked resulting in a 0 (zero) mark for this coursework.
NOTE 1: You are advised to develop and present your software as three separate solutions, even if this means that you'll end up reusing parts of the solutions from previous tasks. If you choose to deliver one piece of software with a menu of choices of running tasks and this fails to run for whatever reason, the marker will NOT try to edit your code to see which parts actually work.

Section 4: Marking Criteria

The following table (please see next page) gives details of the marking criteria for this coursework.
Marks will be awarded for clear rationale justifying design choices.
Clarity in the pseudo code submitted allowing to map the full logic of the solution implemented is expected.
Code must be well structured, appropriately commented, neat and efficient. Clear use of functions and reduced repetitions of blocks of code are expected.

The use of GUI or other user interface will not attract any specific marks, but simplicity and efficiency of its design will be considered when awarding for an overall efficient system developed.

NOTE – No hard coded data will be allowed. Hard coded data in the submitted work will result in the work marked at 0 (zero).

Section 5: Feedback mechanisms

Formative / Verbal will be provided during practical sessions.
Written feedback will be provided on blackboard along with the marked for the submitted work on 24 May 2021.

Marking Criteria Table

	0-29	30-39	40-49	50-59	60-69	70-84	85-100	Mark & Advice for Improvement
Task 1 **35%**	Pseudo code lacks clarity / is incomplete. The program does not deliver a complete solution.	The program works but the constraints are not met in the order they have been specified. Pseudo code and design diagrams show that about half the required features are met.	Pseudo code addresses 51 to 60% of required features. The program runs observing up to three constraints.	Pseudo code addresses 61 to 70% of required features. The program runs observing up to three constraints, with at least the first two in the correct order for the task.	Pseudo code addresses 71 to 80% of required features. The program runs observing up to three constraints, in the correct order for the task.	Pseudo code addresses over 80% of required features. The program runs observing all constraints, in the correct order for the task.	The design delivers all of the required features and goes beyond the requirements in such a way as to propose a solution that is fully efficient and will result in elegant program code.	
Task 2 **50%**	Pseudo code lacks clarity / is incomplete. The program does not deliver a complete solution, struggling to run.	The Pseudo code and design diagrams show that about half the required features are met. The code runs with some issues delivering about 50% of the expected results	Pseudo code addresses 51 to 60% of required features.	Pseudo code addresses 61 to 70% of required features.	Pseudo code addresses 71 to 80% of required features.	Pseudo code addresses over 80% of required features.	The design delivers all of the required features and goes beyond the requirements in such a way as to propose a solution that is fully efficient and will result in elegant program code.	
Task 3 **15%**	The justification of the design choices is minimal at best. The commentary on personal data lacks clarity.	The two pieces of text show minimal clarity, with only one of the items providing some logical points of discussion.	Both points are discussed by the discussion is not conclusive.	At least one of the points of discussion offers a good structural logic that leads to a conclusion.	Both items are discussed in a clear way. Not all facts stated though are fully correct.	Both items are discussed well, there is clarity in the points raised. There is some support by references to the personal data topic. NOTE – References do not count towards the word count.	An excellent discussion, fully evidenced and very well supported by relevant references.	