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 & Mr. James Lear
Year	2019-20
Component/Element Number	B / 1
Total number of assessments for this module	2
Weighting	40%
Element Description	Coursework A

Dates

Date issued to students	21 October 2019
Date to be returned to students	
Submission Date	21 November 2019
Submission Place	Blackboard - Online
Submission Time	14:00
Submission Notes	Please submit all files in a zipped folder labelled "DADSA1920A – Your Name"

Feedback

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

Contents

Module Details 1
Dates 1
Feedback 1
Contents 2
Section 1: Overview of Assessment 3
Section 2: Task Specification 3
Section 3: Deliverables 4
Section 4: Marking Criteria 5
Section 5: Feedback mechanisms 5
Marking Criteria 6

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 ***20%*** 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.

Section 2: Task Specification

The case of an animal sanctuary

An animal sanctuary caters for the protection of pets and animals usually found in the wild. The usual pets that it caters for are dogs and cats while it provides shelter for birds (canaries and parrots).
The wild animals are a collection that varies with the season, most usual are foxes and badgers, but there have been monkeys, deer and occasionally lions and tigers.
The reason for taking in pets is when they are found stray (very rare), when their owner is taken ill or dies and have no one to take of them, found injured after an accident and owner cannot be identified, or the owner cannot continue to take care of them for other reasons and has abandoned them. In all cases the sanctuary tries to identify the owner or if not possible, to find new homes for them. In the case of dogs and cats where they are considered stray they are vaccinated, neutered and microchipped to conform to the legislation.
The wild animals are usually found where they have escaped from a zoo, have been involved in an accident, or the owners cannot continue to take care of them. Depending on the type of animal the sanctuary will aim to release them into the wild after treating them, or in the case of animals that have been kept in a zoo, arrange for their safe transfer to their original owners or to another zoo where it is necessary.

In all cases upon arrival to the sanctuary the animals are given a proper health check and are assigned a sanctuary identification. The vets that examine the animals will decide on what treatment they require – surgery, medication, or both. They also decide on the type of food they will be provided for and the initial duration of stay until declared healthy for the next step in preparing them to leave the sanctuary.
To manage its daily activities the sanctuary maintains records of the animals it caters for. Data is also kept to manage the incomings and outgoings of animals. Data includes sanctuary identification (ID), name, animal type, breed (wherever applicable), date admitted, date left, destination, destination address. Also (where applicable) data I held on surgery performed, medication and microchip number. Samples of data is shown in the attached CSV files.
Assume that no data is ever deleted from the sanctuary's records.
You are required to complete the following tasks
Task 1 – Address the following design considerations

- Identify and create suitable data structures that can be used to store such data.
- Explain your choice of data structures relevant to the type of data stored and the operations that you are required to perform, based on the assumption that the sanctuary admits and treats more than 10000 animals per year.
- Provide detailed pseudocode for the software you have implemented

Task 2 – choose a relevant algorithm and implement the software that will allow the user to do the following subtasks

- A – create an entry for a new arrival
- B – Find the full data for an animal by using the animals sanctuary identification
- C – produce a list of identified people that have abused animals in the past (in alphabetical order)
- D – produce a list of people that have abandoned their animals before (in alphabetical order)
- E – produce a list of cats that are ready for adoption (neutered, vaccinated and microchipped)
- F - produce a list of dogs that are ready for adoption (neutered, vaccinated and microchipped)
- G – Produce a list of animals that are ready to be returned to their owners. The list should comprise both pets and wild animals with all pets listed first and all wild animals following. The listing will be based on sanctuary identification in ascending order

Task 3 – create software that allows the user to edit the data stored

- A – enter details of surgery
- B – enter details of neutering
- C – enter microchip number
- D – edit the status of an animal (please data provided)
- E – edit date of departure from the sanctuary
- F – edit destination of the animal upon departure

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.
The folder must contain:
All program files saved in Python format (3.6 or 3.7) – any other version will not be accepted and the work will not be marked resulting in a 0 (zero) mark for this coursework. (Tasks 2 & 3)
One word or PDF file containing the justification for the choice of data structures (Task 1)
One word or PDF document containing the Pseudo code (Task 1)

One text file with simple instructions of how to use / run your software

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 and during the demonstration sessions after submission of the work.
Written feedback will be provided on blackboard along with the return of the marked work on 9 January 2020.

Marking Criteria Table

	0-29	30-39	40-49	50-59	60-69	70-84	85-100	Mark & Advice for Improvement
Task 1	Choice of Data Structures appears random. Pseudo code lacks clarity / is incomplete.	There is vague attempt to justify choice of data structures. Pseudo code shows logic that does not address all requirements	Pseudo code addresses 51 to 60% of required features as specified in the requirements of Tasks 2 & 3	Pseudo code addresses 61 to 70% of required features as specified in the requirements of Tasks 2 & 3	Pseudo code addresses 71 to 80% of required features as specified in the requirements of Tasks 2 & 3	Pseudo code addresses over 80% of required features as specified in the requirements of Tasks 2 & 3. The design demonstrates elements of efficiency in the solution.	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	Some of the subtasks are met successfully. The code is not neat and comments are sparse and unclear	50% of the expected functionality is delivered, but the successful implementation lacks evidence of intelligence and efficiency.	51 to 60% of the required functionality has been delivered, but issues with code structure, intelligence and efficiency persist	More sub tasks deliver the required functionality but this is no more than 70% of the expected. Code structure and comments leave room for quite a lot of improvement.	The functionality delivered has reached 80 % of the required level. The code appears more neat useful comments have been included in the code.	At least 90% of the code works efficiently and delivers the required functionality.	All tasks have been fully met. The code is elegant, well documented and efficient. The software exceeds the requirements offering a more complete and intelligent solution.	
Task 3	Some of the subtasks are met successfully. The code is not neat and comments are sparse and unclear	50% of the expected functionality is delivered, but the successful implementation lacks evidence of intelligence and efficiency.	51 to 60% of the required functionality has been delivered, but issues with code structure, intelligence and efficiency persist	More sub tasks deliver the required functionality but this is no more than 70% of the expected. Code structure and comments leave room for quite a lot of improvement.	The functionality delivered has reached 80 % of the required level. The code appears more neat useful comments have been included in the code.	At least 90% of the code works efficiently and delivers the required functionality.	All tasks have been fully met. The code is elegant, well documented and efficient. The software exceeds the requirements offering a more complete and intelligent solution.	